PTQ-PDPMV1

Quantum/Unity Platform

PROFIBUS DP V1 Master

Application Reference Guide

May 30, 2007

Please Read This Notice

Successful application of this module requires a reasonable working knowledge of the Schneider Electric Quantum hardware, the PTQ-PDPMV1 Module and the application in which the combination is to be used. For this reason, it is important that those responsible for implementation satisfy themselves that the combination will meet the needs of the application without exposing personnel or equipment to unsafe or inappropriate working conditions.

This manual is provided to assist the user. Every attempt has been made to assure that the information provided is accurate and a true reflection of the product's installation requirements. In order to assure a complete understanding of the operation of the product, the user should read all applicable Schneider Electric documentation on the operation of the Schneider Electric hardware.

Under no conditions will ProSoft Technology be responsible or liable for indirect or consequential damages resulting from the use or application of the product.

Reproduction of the contents of this manual, in whole or in part, without written permission from ProSoft Technology is prohibited.

Information in this manual is subject to change without notice and does not represent a commitment on the part of ProSoft Technology Improvements and/or changes in this manual or the product may be made at any time. These changes will be made periodically to correct technical inaccuracies or typographical errors.

Important Notice:

CAUTION: THE CELL USED IN THIS DEVICE MAY PRESENT A FIRE OR CHEMICAL BURN HAZARD IF MISTREATED. DO NOT DISASSEMBLE, HEAT ABOVE 100°C (212°F) OR INCINERATE.

Maximum battery load = 200 µA.

Maximum battery charge voltage = 3.4 VDC.

Maximum battery charge current = 500 µA.

Maximum battery discharge current = 30 µA.

Your Feedback Please

We always want you to feel that you made the right decision to use our products. If you have suggestions, comments, compliments or complaints about the product, documentation or support, please write or call us.

ProSoft Technology
1675 Chester Avenue, Fourth Floor
Bakersfield, CA 93301
+1 (661) 716-5100
+1 (661) 716-5101 (Fax)
http://www.prosoft-technology.com

Copyright © ProSoft Technology, Inc. 2000 - 2007. All Rights Reserved.

PTQ-PDPMV1 Application Reference Guide
May 30, 2007
PSFT.PDPMV1.PTQ.UM.07.05.30

ProSoft Technology ®, ProLinx ®, inRAx ®, ProTalk® and RadioLinx ® are Registered Trademarks of ProSoft Technology, Inc.

Contents

PLEASE READ THIS NOTICE 2
Important Notice: 2
Your Feedback Please 2

GUIDE TO THE -PDPMV1 USER MANUAL 7

1 START HERE 9
1.1 Deployment Checklist 9
1.2 Install ProSoft Configuration Builder Software 10
1.3 Install the Module 10
1.3.1 Install the ProTalk Module in the Quantum Rack 10
1.3.2 Connect to the ProTalk Configuration/Debug Port 11
1.3.3 PTQ-PDPMV1 Configuration / Debug Port Note 13

2 CONFIGURING THE MODULE 15
2.1 Configure the Module with ProSoft Configuration Builder 15
2.1.1 Set Up the Project 17
2.1.2 Set Module Parameters 19
2.1.3 Update the Ethernet Settings 25
2.2 Downloading the Ethernet Configuration to the Module 26
2.3 Configure the PROFIBUS Master 27
2.3.1 Install the GSD Files 28
2.3.2 Configure the PROFIBUS Slaves 29
2.3.3 Export the Processor Memory Map 33
2.3.4 Download the Project to the Module 35
2.3.5 Back Up the Project 36
2.4 Configure the Processor 38

3 CONFIGURING THE PROCESSOR WITH UNITY PRO XL 39
3.1 Import the Functional Module 39
3.1.1 Updating Checksum Values - UnityPro XL 44
3.1.2 Setup General Unity Pro XL Project Settings 46
3.1.3 Configure the Memory Size for the Processor 48
3.1.4 Build the Project 50
3.1.5 Download the Project to the Processor 51
3.1.6 Verify Communication between the Processor and the Module 51
3.2 Function Blocks Operation Overview 54
3.3 Derived Function Blocks Overview 57
3.3.1 Using the Derived Function Blocks 57
3.4 Using Mailbox Function Blocks 60
3.4.1 Overview 60
3.4.2 Configuration 60
3.4.3 Trigger (BYTES) 61
3.4.4 Specific Input Pins 62
3.4.5 Specific Output Pins 62
3.5 Mailbox Overview 64
3.5.1 Acyclic Read Mailbox 64

3.5.2 Acyclic Write Mailbox ... 65
3.5.3 Alarm Mailbox ... 65
3.5.4 GetConfiguration Mailbox ... 67
3.5.5 GetDiagnostics Mailbox ... 67
3.5.6 GetLiveList Mailbox ... 68
3.5.7 SetSlaveAddress Mailbox ... 69
3.5.8 SetOperatingMode Mailbox ... 69
3.5.9 SetSlaveMode Mailbox ... 70
3.5.10 StartStopSlaves Mailbox ... 70
3.5.11 Coldboot Mailbox ... 71

4 CONFIGURING THE PROCESSOR WITH CONCEPT ... 73
4.1 Overview ... 73
4.2 Before You Begin ... 74
4.3 Information for Concept Version 2.6 Users ... 75
4.3.1 Installing MDC Configuration Files ... 75
4.4 Step 1 - Exporting the Files from PCB ... 76
4.4.1 -.ASC files ... 77
4.4.2 -.DTY file ... 78
4.4.3 -.TXT file ... 78
4.5 Step 2 - Convert the Function Blocks ... 78
4.6 Step 3 - Setup the Concept Project ... 81
4.7 Step 4 - Import the Variables ... 83
4.8 Step 5 - Create the Function Block Instances ... 86
4.9 Step 6 - Download the Concept Project ... 92
4.10 Using the Concept Project ... 92
4.10.1 Accessing PROFIBUS Data ... 92
4.10.2 Accessing Status Data ... 93
4.10.3 Configuration Validation & SETCRC Function Block ... 94
4.11 Using Mailbox Function Blocks ... 95
4.11.1 Overview ... 95
4.11.2 Configuration ... 95
4.11.3 Trigger Register ... 96
4.11.4 Specific Input Pins ... 97
4.11.5 Specific Output Pins ... 97
4.12 Mailbox Overview ... 100
4.12.1 Acyclic Read Mailbox ... 100
4.12.2 Acyclic Write Mailbox ... 101
4.12.3 Alarm Mailbox ... 102
4.12.4 GetConfiguration Mailbox ... 103
4.12.5 GetDiagnostics Mailbox ... 103
4.12.6 GetLiveList Mailbox ... 104
4.12.7 SetSlaveAddress Mailbox ... 105
4.12.8 SetOperatingMode Mailbox ... 105
4.12.9 SetSlaveMode Mailbox ... 106
4.12.10 Start/Stop Slaves Mailbox ... 106
4.12.11 Coldboot Mailbox ... 107

5 CONFIGURING THE PROCESSOR WITH PROWORX 32 ... 109

6 MAILBOX MESSAGING ... 113
6.1 Mailbox Message Queuing ... 113
6.1.1 Queue Timeouts ... 114
6.2 Special Function Mailbox Messaging Commands ... 114

6.2.1 Mailbox Message - Set Slave Mode 116
6.2.2 Mailbox Message - Get Slave Diagnostics 119
6.2.3 Mailbox Message - Get Slave Configuration 121
6.2.4 Mailbox Message - Set Slave Address 122
6.2.5 Mailbox Message - Get Live List 124
6.2.6 Mailbox Message - Acyclic Data Read - Class 1 126
6.2.7 Mailbox Message - Acyclic Data Write - Class 1 127
6.2.8 Mailbox Message - Alarm Indication 129
6.2.9 Mailbox Message - Set Operating Mode 131
6.3 Receiving Mailbox Message Responses from module 132
6.4 Mailbox Messaging Error Codes 134
6.4.1 Message Information 134
6.4.2 Return Codes 134
6.4.3 Error Codes 135
6.4.4 DPV1 Error Codes 136

7 HOT STANDBY SUPPORT 137
7.1 Hot Standby Overview 137
7.1.1 Identical Configurations 137
7.1.2 Primary and Standby Controllers 138
7.1.3 System Components 138
7.1.4 Modicon Quantum Hot Standby with Unity and IEC Logic 139
7.1.5 Understanding System Scan Time in Modicon Quantum Hot Standby with Unity Systems 139
7.2 Setting Up the Modicon Quantum Hot Standby with Unity System 142
7.2.1 Overview 142
7.2.2 Mapping the Backplane Extensions 143
7.2.3 PTQ-PDPMV1 Hot Standby Considerations 143
7.2.4 Hot Standby States 144
7.2.5 Transition Description 145
7.2.6 HSBY State vs. Master Operation Mode 146
7.2.7 Ping Message 146
7.2.8 PTQ Link Message 150
7.2.9 Crossed Status Information 150
7.2.10 Conditions for Switchover 151
7.3 PTQ-PDPMV1 Operation 152
7.3.1 PTQ-PDPMV1 HSBY Diagnostic Data 152
7.3.2 Time for Switchover 162
7.3.3 Bus Parameters 162
7.3.4 HSBY Master GSD-file 163
7.3.5 ProSoft Configuration Builder (PCB) HSBY Option Functionality 164
7.4 PTQ-PDPMV1 Master Bus Properties when using a P&F DP/PA Segment Coupler 166
7.4.1 PROFIBUS DP time behavior 166
7.4.2 Commissioning of communication with the SK1 segment coupler 167
7.4.3 Commissioning of communication with the SK2 segment coupler 169
7.4.4 Details for calculating the TWD parameter 171

8 DIAGNOSTICS AND TROUBLESHOOTING 173
8.1 Basic Troubleshooting Steps 173
8.2 LED Status Indicators - Front of PTQ module 174
8.3 Module Status Indicators 177
8.4 PROFIBUS Master Indicators 178
8.5 The Configuration/Debug Menu 178

8.5.1 Using the Diagnostic Window in ProSoft Configuration Builder 178
8.5.2 Navigation 179
8.5.3 Main Menu 180
8.5.4 Input Data View Menu 185
8.5.5 Output Data View Menu 186

9 REFERENCE 189
9.1 Product Specifications 189
9.1.1 Features and Benefits 191
9.1.2 Hot Standby Support 191
9.1.3 General Specifications 191
9.1.4 Hardware Specifications 192
9.1.5 Functional Specifications 192
9.2 Functional Overview 193
9.2.1 About the PROFIBUS Protocol 193
9.2.2 General Overview 193
9.2.3 PROFIBUS DP Architecture 194
9.2.4 Master/Slave Communication Phases 195
9.2.5 PTQ Input and Output Data Blocks 196
9.3 PROFIBUS comDTM 207
9.3.1 ProSoft Technology product availability 207
9.3.2 Introduction 207
9.3.3 Hardware Requirements (Recommended) 210
9.3.4 Software Requirements (Minimum) 210
9.4 Cable Connections 217
9.4.1 Ethernet Connection 217
9.4.2 RS-232 Configuration/Debug Port 218
9.5 PROFIBUS Master Port 218
9.6 Constructing a Bus Cable for PROFIBUS DP 218
9.7 Supported PROFIBUS Services 223
9.8 Quantum to PTQ Communication Protocol 224
9.9 Calculating System Response Time 225
9.9.1 How to Calculate PROFIBUS Time – TMC4 226
9.9.2 Calculating System Reaction Time 227
9.10 Using Multiple PTQ-PDPMV1 Modules with Concept 228
9.11 Frequently Asked Questions 229
9.11.1 How do I configure the module? 229
9.11.2 Is a .MDC available for configuration of the Module? 229
9.11.3 Does the module work in a remote rack? 229
9.11.4 Can I use the module in a hot backup system? 229

SUPPORT, SERVICE & WARRANTY 231
Module Service and Repair 231
General Warranty Policy – Terms and Conditions 232
Limitation of Liability 233
RMA Procedures 233

INDEX 235

Guide to the -PDPMV1 User Manual

Function		Section to Read	Details
Introduction (Must Do)	→	***Start Here*** (page 9)	This Section introduces the customer to the module. Included are: package contents, system requirements, hardware installation, and basic configuration.
Verify Communication, Diagnostic and Troubleshooting	→	***Verifying Communication*** (page 174) ***Diagnostics and Troubleshooting*** (page 172)	This section describes how to verify communications with the network. Diagnostic and Troubleshooting procedures.
Reference Product Specifications Functional Overview Glossary	→	***Reference*** (page 188) ***Functional Overview*** (page 193) ***Product Specifications*** (page 189)	These sections contain general references associated with this product, Specifications, and the Functional Overview.
Support, Service, and Warranty Index	→	***Support, Service and Warranty*** (page 230)	This section contains Support, Service and Warranty information. Index of chapters.

1 Start Here

In This Chapter

- Deployment Checklist 9
- Install ProSoft Configuration Builder Software 10
- Install the Module 10

1.1 Deployment Checklist

This is a list of the steps you must complete to install your ProTalk Module. We recommend that you read through this section completely before you begin the installation.

During this procedure, you will install the module in the rack with the processor, set up a PROFIBUS master, connect one or more PROFIBUS slave devices, and then configure the processor with information about the PROFIBUS network. The example programs you will be configuring are designed to demonstrate that the module and the processor are correctly configured and communicating with each other over the backplane. After this initial installation, you may need to perform additional steps to configure the application for your specific needs.

You must complete these steps in the following order, otherwise the installation may not be successful.

1 Install the ProSoft Configuration Builder software on your PC.

Important: Earlier versions of ProSoft Configuration Builder do not support the Hot Standby (HSBY) feature on the PTQ-PDPMV1 module. To make full use of the HSBY feature, please download the latest version of ProSoft Configuration Builder and review the readme files from the ProSoft Technology web site at http://www.prosoft-technology.com/pcb.

2 Install the ProTalk module in the rack.
3 Configure the Module.
4 Configure the PROFIBUS Master and Slaves
5 Export the Processor Files
6 Configure the Processor
7 Verify communication between the processor and the module.

1.2 Install ProSoft Configuration Builder Software

You must install the ProSoft Configuration Builder (PCB) software in order to configure the PTQ-PDPMV1 module. Go to the ProSoft Technology web site at http://www.prosoft-technology.com/pcb for the latest version of PCB.

➢ *To install ProSoft Configuration Builder:*

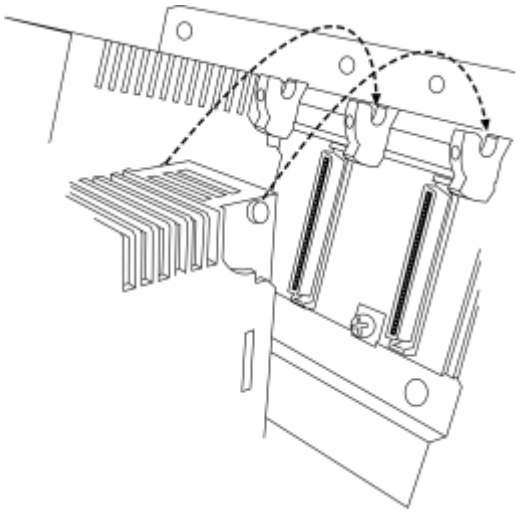
1. Insert the ProSoft Configuration Builder CD-ROM into the CD drive of your PC. Wait for the startup screen to appear.
2. On the startup screen, click *Product Documentation*. This action opens an explorer window.
3. Click to open the *Utilities* folder. This folder contains all of the applications and files you will need to set up and configure your module.
4. Double-click the *ProSoft Configuration Builder Setup* program and follow the instructions on your screen to install the software on your PC.

Note: Many of the configuration and maintenance procedures use files and other utilities on the CD-ROM. You may wish to copy the files from the Utilities folder on the CD-ROM to a convenient location on your hard drive.

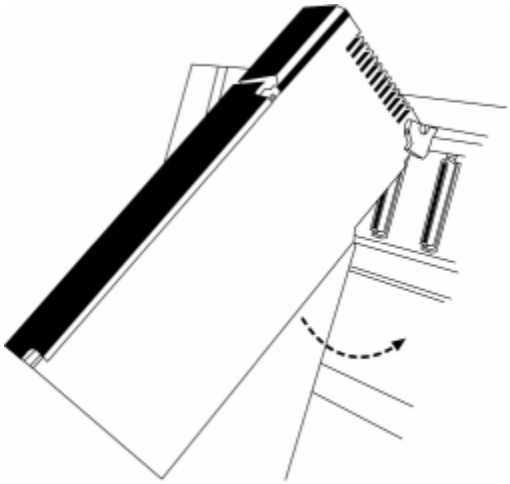
1.3 Install the Module

1.3.1 Install the ProTalk Module in the Quantum Rack

1. Place the Module in the Quantum Rack. The ProTalk module must be placed in the same rack as the processor.
2. Tilt the module at a 45° angle and align the pegs at the top of the module with slots on the backplane.



3 Push the module into place until it seats firmly in the backplane.



CAUTION: The PTQ module is hot-swappable, meaning that you can install and remove it while the rack is powered up. You should not assume that this is the case for all types of modules unless the user manual for the product explicitly states that the module is hot-swappable. Failure to observe this precaution could result in damage to the module and any equipment connected to it.

HSBY Note: For HSBY setup repeat the above procedures for the Primary and Standby modules.

1.3.2 Connect to the ProTalk Configuration/Debug Port

Note: The module has a serial port as well as an Ethernet port. The first time you connect to the module to configure it, you can connect to the module's serial port using the supplied null-modem cable, because the module's default Ethernet settings may not match your network.

HSBY Note: For HSBY units the Ethernet connection must be applied. This connection is used as a backup to ping status messages over the PROFIBUS network. It is also used for DPV1 passive master buffer update during switchover.

PC to Ethernet Port Connection

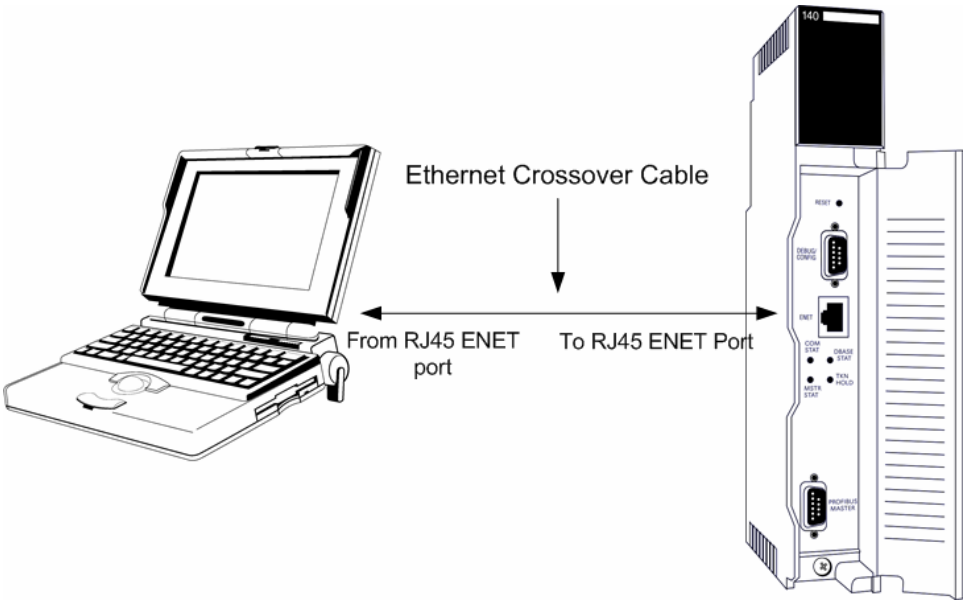
Important: The PTQ-PDPMV1 module is equipped to use an Ethernet connection using the following defaults:

```
My_ip:    192.168.0.100
Netmask:  255.255.255.0
Gateway:  192.168.0.1
```

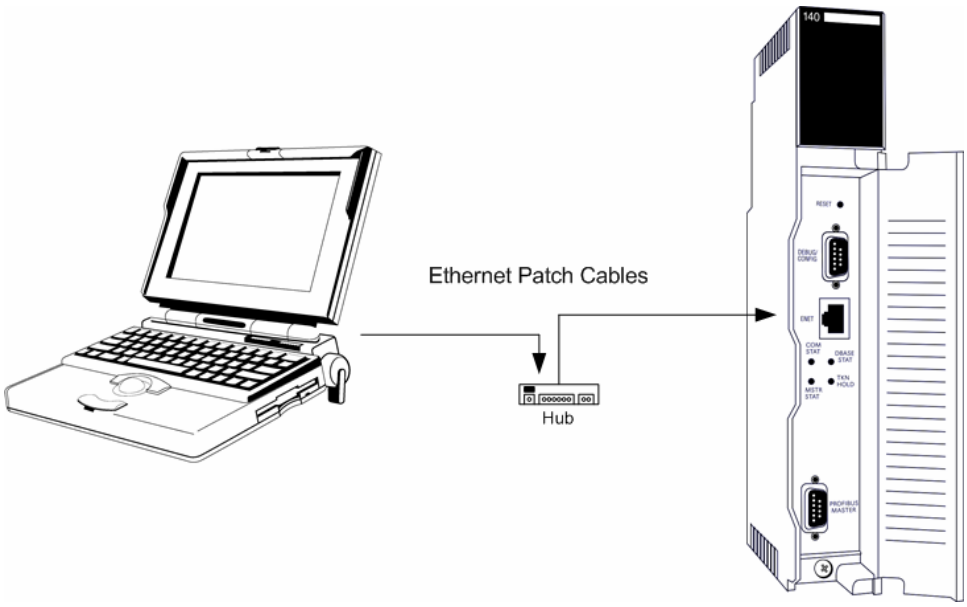
HSBY Note: For HSBY units the passive master module Ethernet connection is always Primary IP plus 1. For example, Primary IP = 192.168.0.100, Standby module IP = 192.168.0.101. This is not controlled by user; the modules firmware automatically sets the IP address of the passive master.

If you cannot use these defaults for your connection, you must change them using ProSoft Configuration Builder and then download the new values to the PTQ-PDPMV1 module via serial cable or directly using a Compact Flash (CF) writer. If you plan on changing the Ethernet addresses, use ProSoft Configuration Builder to change the values in the WATTCP file.

If the default values are OK, and you are using an Ethernet connection, please connect your PC to the PTQ-PDPMV1 module using either of the methods described below:



PC to Ethernet Port Connection via Hub



1.3.3 PTQ-PDPMV1 Configuration / Debug Port Note

After the Ethernet settings are correctly configured, only the Ethernet port should be used for configuration changes, diagnostics and PROFIBUS monitor.

2 Configuring the Module

In This Chapter

- Configure the Module with ProSoft Configuration Builder 15
- Downloading the Ethernet Configuration to the Module 25
- Configure the PROFIBUS Master 27
- Configure the Processor 38

2.1 Configure the Module with ProSoft Configuration Builder

In this step of the setup process, you will use ProSoft Configuration Builder to configure the parameters that affect the interface between the PTQ module and the processor (Quantum or Unity). These parameters indicate:

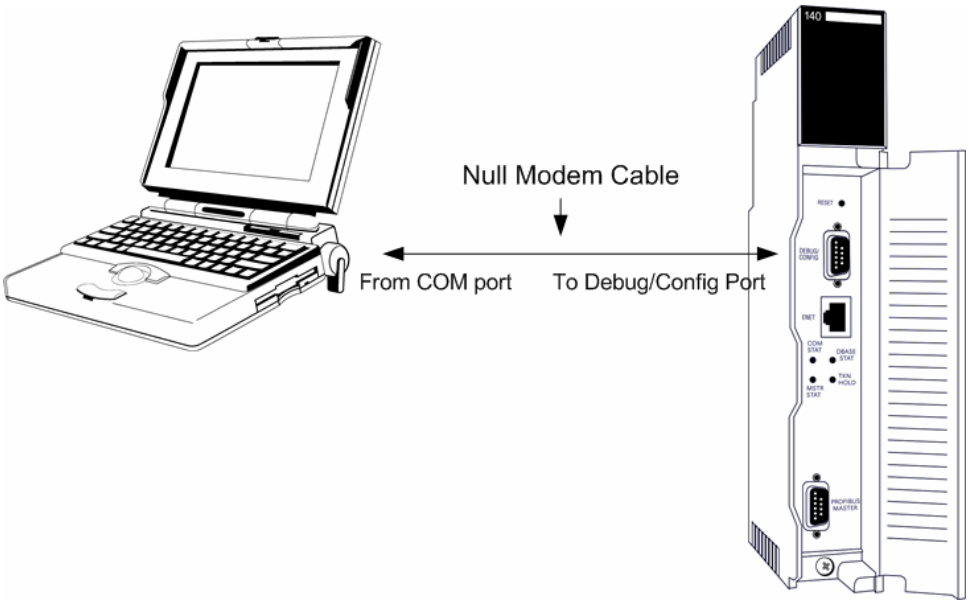
- The physical position of the module in the rack.

HSBY Note: For HSBY units, the active and passive modules must be placed in the same rack location in both racks.

- The starting memory address in the processor's State RAM for the module's input and output data images. For the purpose of this example, we use a starting address of 1000 for the input image and 3000 for the output image.

To begin, verify that the processor is correctly positioned in the rack, and is powered up. Connect your PC to the PTQ-PDPMV1 module using the supplied Null Modem serial cable, as shown in the following illustration.

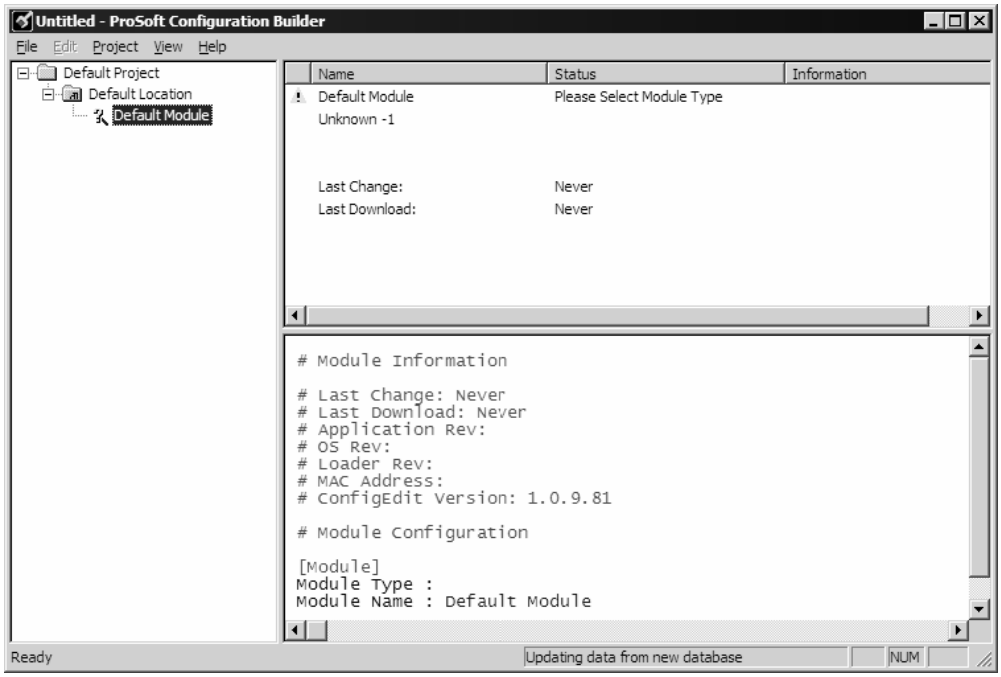
Note: The serial port should only be used for initial configuration of the Ethernet port through ProSoft Configuration Builder.



After the Ethernet settings are correctly configured, only the Ethernet port should be used for configuration changes, diagnostics and PROFIBUS monitor.

2.1.1 Set Up the Project

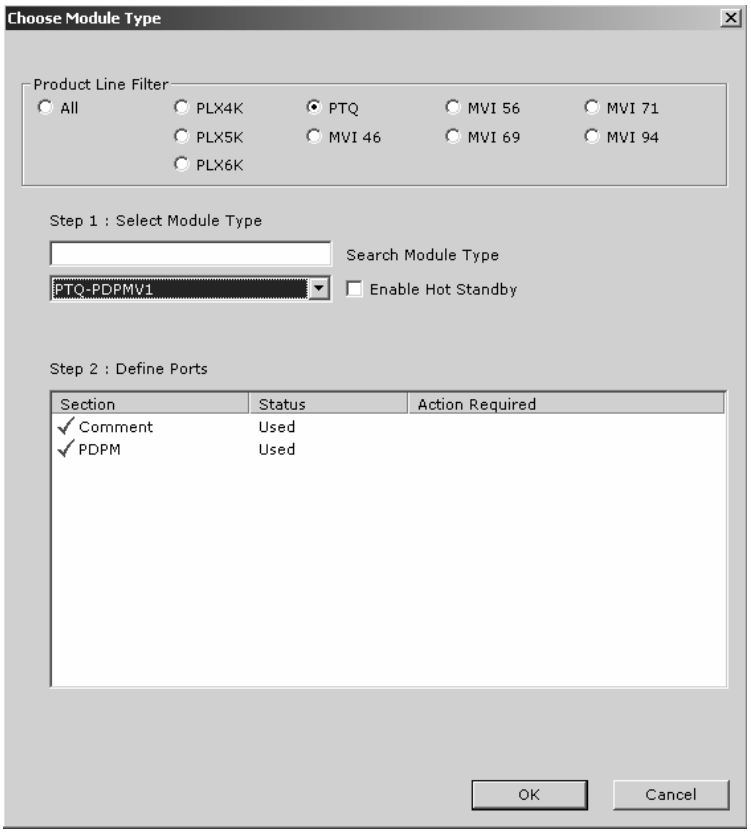
To begin, start ProSoft Configuration Builder. If you have used other Windows configuration tools before, you will find the screen layout familiar. ProSoft Configuration Builder's window consists of a tree view on the left, an information pane and a configuration pane on the right side of the window. When you first start ProSoft Configuration Builder, the tree view consists of folders for Default Project and Default Location, with a Default Module in the Default Location folder. The illustration below shows the ProSoft Configuration Builder window with a new project.



Your first task is to add the PTQ-PDPMV1 module to the project.

1 Use the mouse to select "Default Module" in the tree view, and then click the right mouse button to open a shortcut menu.

2 On the shortcut menu, choose "Choose Module Type". This action opens the Choose Module Type dialog box.



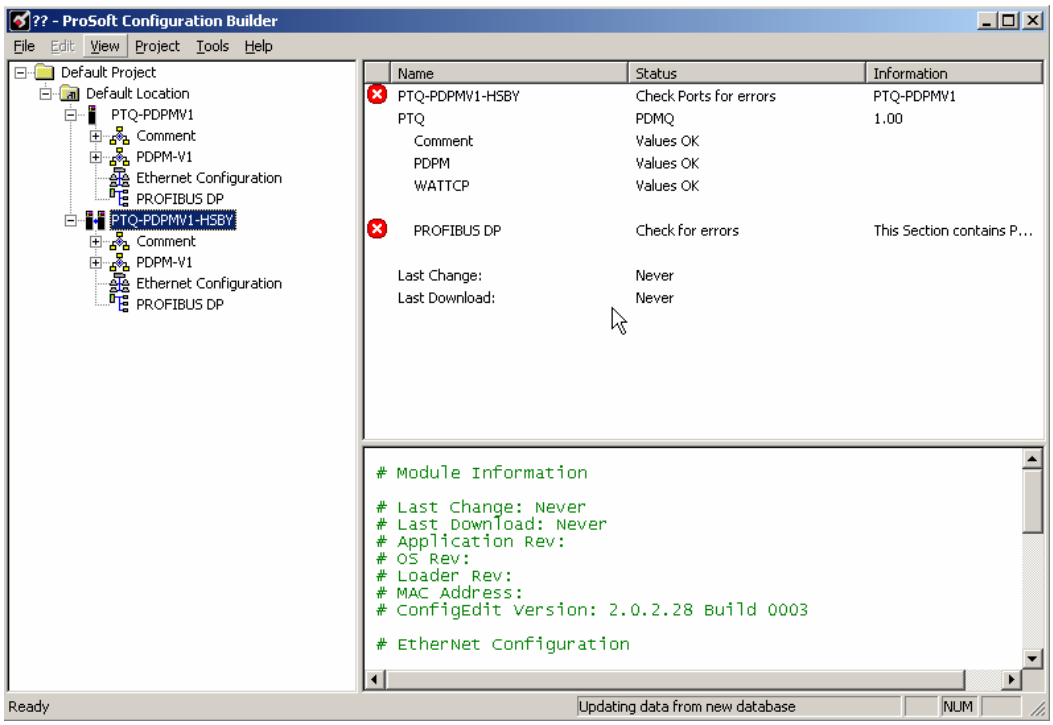
HSBY Note: For Hot Standby support select the Enable "Hot Standby" checkbox.

3 In the Product Line Filter area of the dialog box, select PTQ. In the Select Module Type dropdown list, select PTQ-PDPMV1, and then click OK to save your settings and return to the ProSoft Configuration Builder window.

The next task is to set the module parameters.

2.1.2 Set Module Parameters

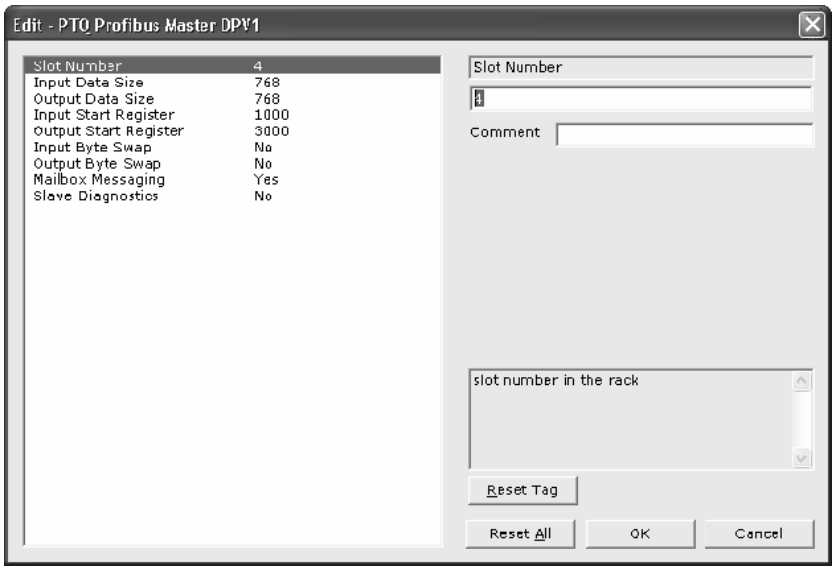
Notice that the contents of the information pane and the configuration pane changed when you added the PTQ-PDPMV1 module to the project. The red "X" icon indicates that the module's configuration is incomplete.



HSBY Note: For Hot Standby modules a double module icon will be displayed

In the following steps, you will provide the missing information to begin configuring the module.

1. Click the plus sign **[+]** next to the module to expand the module tree, and then expand the PDPM-V1 tree.
2. Double-click the PTQ PROFIBUS Master object. This action opens the **Edit** dialog box.
3. In the Edit dialog box, change the values for the selections in this section of the configuration to match the values in the following illustration. To change a value, select the parameter to modify in the left pane, and then type the new value in the edit field in the right pane. If you are not sure what to enter here, use the default values.



Note: The values you enter for the purpose of this example configuration are used by the sample program that you will download to the processor later in this section. You may need to change these values as you implement your production system. Use the following chapters for your Quantum or Unity configuration software, or the online help system, for detailed information on each of the parameters associated with the module.

Slot Number

The Slot Number is the physical location of the module in the rack. The example here assumes a basic configuration with a power supply occupying the first slot, the processor occupying the next two slots, and the PTQ-PDPMV1 module occupying the fourth slot.

The slot number for this example is 4.

Note: If the module is not placed in the slot number specified, the module will not operate, and the CFG ERR light will illuminate. You must specify the actual slot number for the module in the module configuration file.

```
MODULE CONFIGURATION:

PTQ-PDPMV1_MBANDDIAG

Slot Number          : 3       (Found In Slot Number 3)
Profibus Input Size  : 768     (words)
Profibus Output Size : 768     (words)
Input Start Reg      : 1000      Total Input Size  : 1369    (words)
Output Start Reg     : 3000      Total Output Size : 918     (words)
```

Input Data Size

Number of PROFIBUS input point words. Leave this setting at its default value of 768 words.

Output Data Size

Number of PROFIBUS output point words. Leave this setting at its default value of 768 words.

Input Start Register

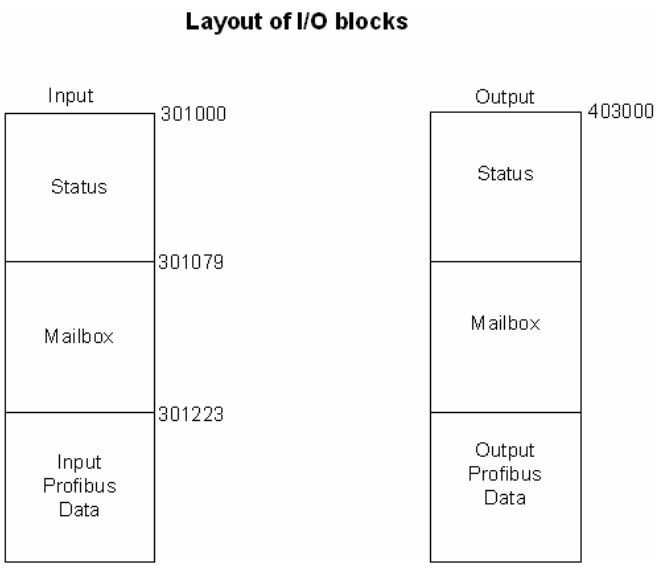
The Input Start Register address refers to the 3x (%IW) location in the processor's State RAM and the Output Start Register refers to 4x (%MW) location of State RAM. You can view State RAM information in Unity XL Pro.

Important: The Input and Output Start Register parameters define the start registers for the input and output blocks that are transferred between the processor and the module. The PROFIBUS I/O associated to the slaves is part of these blocks. Refer to PTQ Input and Output Data Blocks for a description. Each block contains status, PROFIBUS data, and Mailbox/Slave diagnostics, if chosen.

Output Start Register

4x or %MW start register for data from controller

A common mistake is to assume that because the Input Start Register parameter starts at address 301000, then the PROFIBUS data associated with the slaves will also start at the same register. As the diagram above shows, the Input PROFIBUS Data would start at address 301223 for this example.



Refer to ***PTQ Input Data Block*** (page 196) for detailed information on the contents of these blocks, and a discussion of how various configuration options change the layout of these blocks.

Input Byte Swap

Swap bytes in output image (Yes/No). The default value is No.

This is a user configured flag to indicate if input data is swapped before placing in the input image for the controller. If 0, no swapping. If not 0, then swap bytes.

For more information on byte swapping, please refer to ***Input Block Status Data Details*** (page 201).

Output Byte Swap

Swap bytes in output image (Yes/No). The default value is No.

This is a user configured flag to indicate if output data is swapped after receiving from the output image of the controller. If 0, no swapping. If not 0, then swap bytes.

Mailbox Messaging

Use mailbox messaging over the backplane (Y/N with Y=default).

For this example, leave the setting at its default. For more information on the effect of this setting, please refer to ***Mailbox Messaging*** (page 112).

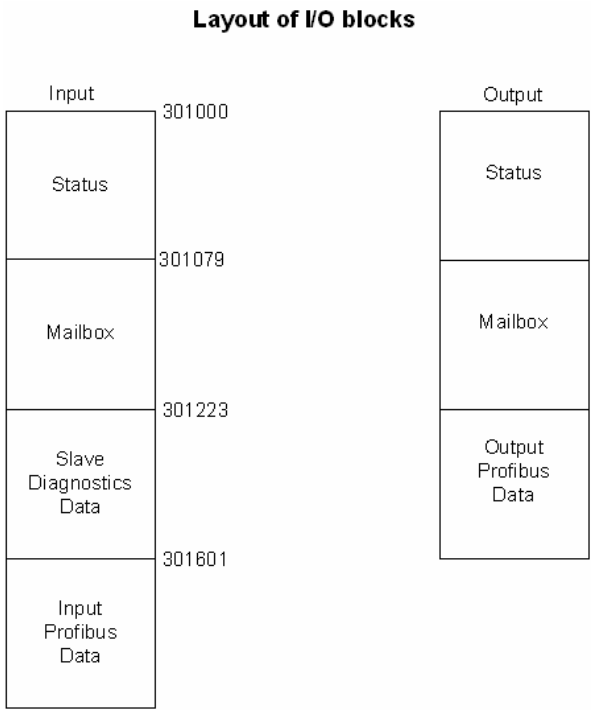
Slave Diagnostics

```
Get slave diagnostic data (Y/N with N=default).
```

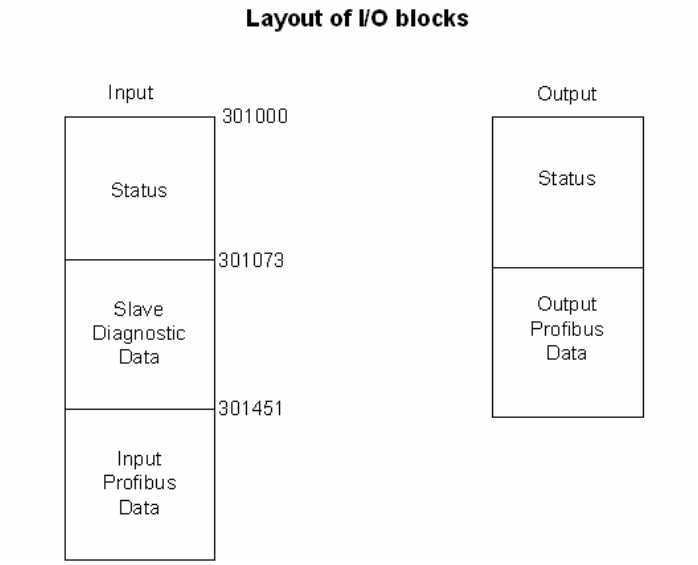
For this example, leave the setting at its default.

If you change the default value of this setting and the previous one (Mailbox Messaging) from their default values, the layout of the I/O blocks changes.

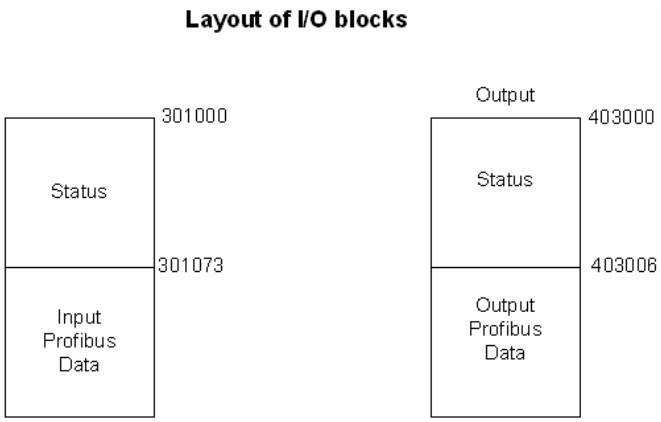
The following diagram shows the layout of the I/O blocks when Mailbox Messaging is set to Yes (the default value), and Get Slave Diagnostic Data is set to Yes.



The following diagram shows the layout of the I/O blocks if you set Mailbox Messaging to No, and Slave Diagnostics to Yes.



The following diagram shows the layout of the I/O blocks if you set Mailbox Messaging to No, and Slave Diagnostics to No.



In ProSoft Configuration Builder, the "Show Concept Map" and "Show Unity Map" commands show the layout of the entire input and output backplane blocks.

Refer to ***PTQ Input Data Block*** (page 196) for detailed information on the contents of these blocks, and a discussion of how various configuration options change the layout of these blocks.

Completing the Example Configuration

When you have finished updating the values, click OK to save your settings and return to the ProSoft Configuration Builder window.

At this time, you may wish to rename the "Default Project" and "Default Location" folders in the tree view. To rename an object:

1. Select the object, and then click the right mouse button to open a shortcut menu. From the shortcut menu, choose Rename.
2. Type the name to assign to the object.
3. Click away from the object to save the new name.

The next task is to update the module's Ethernet settings. This allows you to connect from your PC to the module using an Ethernet cable rather than a serial cable.

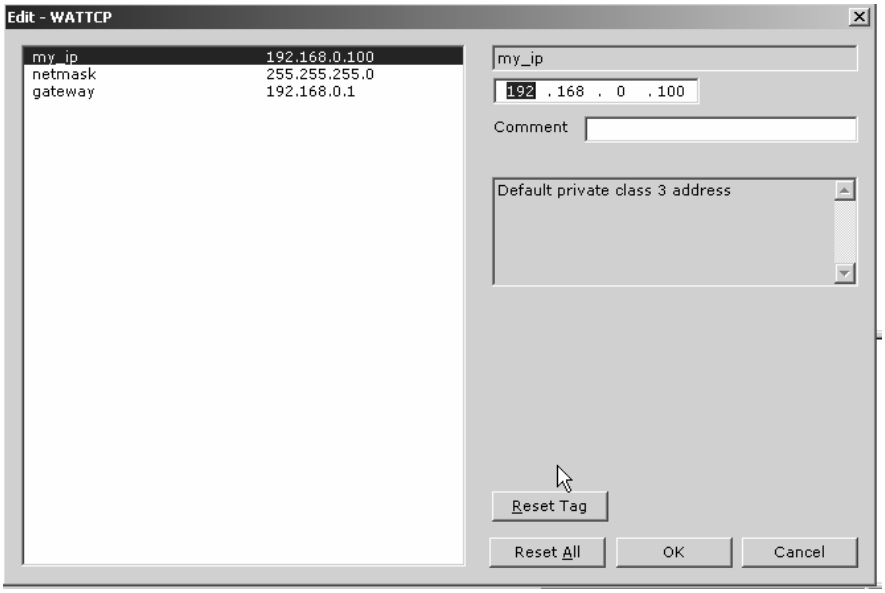
2.1.3 Update the Ethernet Settings

Use this procedure to configure the Ethernet settings for your module. You must assign an IP address, subnet mask and gateway address. After you complete this step, you can connect to the module with an Ethernet cable.

1 Determine the network settings for your module, with the help of your network administrator if necessary. You will need the following information:
   - o IP address (fixed IP required) _____ . _____ . _____ . _____
   - o Subnet mask _____ . _____ . _____ . _____
   - o Gateway address _____ . _____ . _____ . _____

HSBY Note: Hot Standby Primary IP is entered. The Standby IP address will always be the Primary IP address plus 1

2 Click [+] to expand the tree for the PTQ-PDPMV1 module.
3 Double-click the WATTCP object. This action opens the Edit dialog box.



Notice that this is the same dialog box that you used to configure the PTQ PROFIBUS Master in the previous step. The parameters listed here for editing correspond with the object you selected in the tree view.

4 Edit the values for my_ip, netmask (subnet mask) and gateway (default gateway).
5 When you are finished editing, click OK to save your changes and return to the ProSoft Configuration Builder window.

2.2 Downloading the Ethernet Configuration to the Module

In order for your changes to take effect, you must download (copy) the updated Ethernet Configuration from your PC to the module.

1. Connect the serial cable between the module and the PC.
2. Select the Ethernet Configuration icon, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose Download. This action sends the new IP settings to the module, allowing Ethernet communication between the PC and the module.

HSBY Note: This serial download procedure must be performed for both master HSBY modules

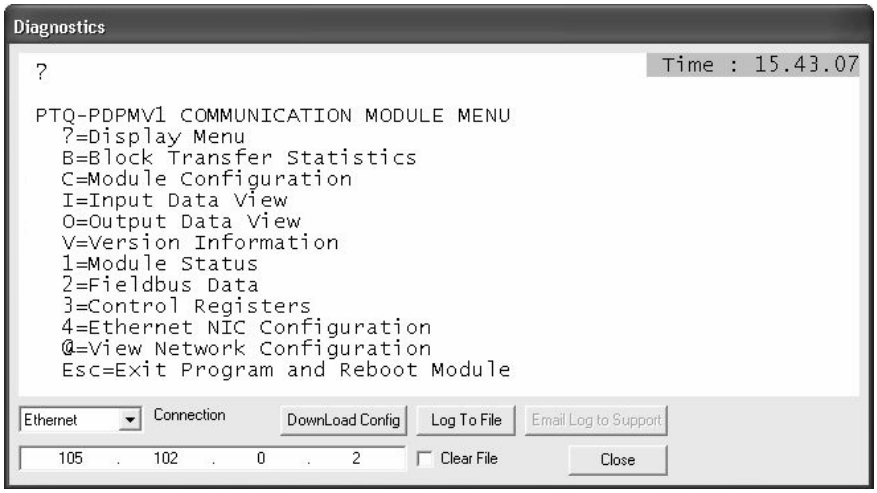
Note: The processor (Quantum or Unity) must be in "Stop" mode before you download the file to the module. Use the processor's softkeys on the display keypad, or the processor's configuration program to stop the processor.

The final step is to verify that ProSoft Configuration Builder can communicate with the module using an Ethernet connection.

1. Plug in an Ethernet cable between the module and an Ethernet hub or router.

HSBY Note: You must leave the Ethernet cable connected to both Hot Standby modules at all times. The configuration download will not proceed unless both modules are connected.

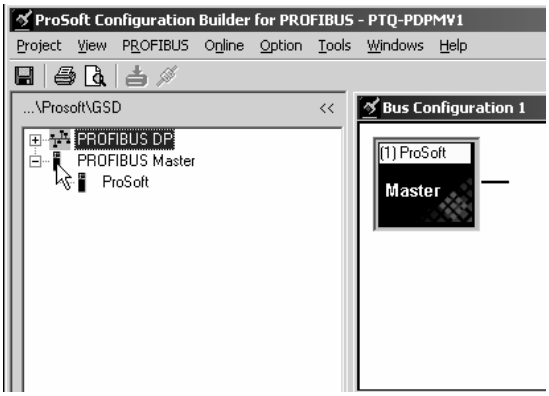
2. In the tree view in ProSoft Configuration Builder, click once to select the PTQ-PDPMV1 module.
3. Open the **Project menu,** and then choose **Module,** and then choose **Diagnostics.** This action opens the Diagnostics window.
4. Choose Ethernet as the connection type, and then enter the IP address. Press the **[?]** key on your keyboard. If the module is communicating successfully, you will see a menu like this:



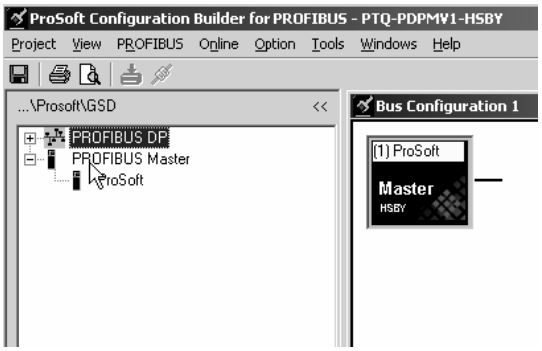
2.3 Configure the PROFIBUS Master

In this task, you will configure the PROFIBUS Master, and then add PROFIBUS Slaves to the network. When this step is complete, you will download the configuration information to the PTQ module. You will also export the I/O maps for the processor.

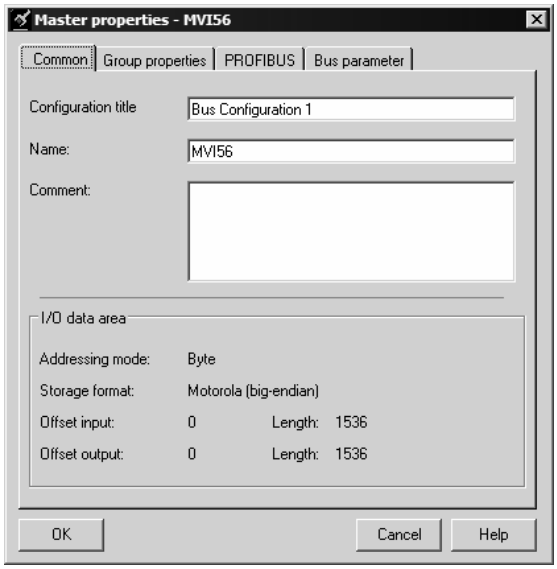
1. In ProSoft Configuration Builder tree view, click **[+]** to expand the PTQ-PDPMV1 tree, and then double-click the PROFIBUS DP icon. This action opens the PROFIBUS Master Setup dialog box.
2. Click the Configure PROFIBUS button. This action opens the ProSoft Configuration Builder for PROFIBUS application.
3. Click [+] to expand the PROFIBUS Master tree.
4. Drag the ProTalk icon into the Bus Configuration window. This is automatically done by the software for new applications.



For HSBY Units



5 Double-click the PROFIBUS Master icon in the Bus Configuration window. This action opens the Master Properties dialog box.



- On the Common tab, name your PROFIBUS drop. The name should match the module name from step 4 in this procedure.

6 On the PROFIBUS tab, select the Address, Baud rate, and the Profile.

- Address is the PROFIBUS address for the module. The default value is 1.
- Baud rate is the speed at which the module can send and receive data. The default value is 1500 kBit/sec.
- Profile: The correct profile setting is "DP" as a "Single Master"

HSBY Note: The correct profile setting for HSBY Master is "DP" however the Hot Standby check box will be checked. The minimum baud for Hot Standby module to switchover within 300 ms with an average processor scan time of 100ms, is 1500Kbits/second

7 Click OK to save your changes and return to the Bus Configuration window. The next step is to install the GSD configuration files for the PROFIBUS slaves you will be adding to the network.

2.3.1 Install the GSD Files

The GSD configuration files contain information on PROFIBUS slaves that you can configure as part of your PROFIBUS network. In order for this configuration information to be available in ProSoft Configuration Builder, you must install the GSD files.

Tip: GSD configuration files for popular Schneider Electric and ProSoft Technology modules are included with the installation. If you have other GSD files for your PROFIBUS slaves, copy them into C:\Documents and Settings\All Users\Application Data\ProSoft\GSD (Windows XP / 2000) or C:\My Documents\ (Windows 98) and ProSoft Configuration Builder will load them automatically.

➢ ***To install GSD files manually:***

1. In ProSoft Configuration Builder tree view, click [+] to expand the PTQ-PDPMV1 tree, and then double-click the PROFIBUS DP icon. This action opens the PROFIBUS Master Setup dialog box.
2. Click the Configure PROFIBUS button. This action opens the ProSoft Configuration Builder for PROFIBUS application.
3. Click [+] to expand the PROFIBUS DP tree.
4. Click the right mouse button to open a shortcut menu.
5. On the shortcut menu, choose "Install new GS* file". This action opens a dialog box that allows you to browse for the location of the GSD configuration files to install.
6. Choose the file to install, and then click Open. If the file already exists in the configuration file path (see Tip above), you will be prompted to overwrite the file.
7. You will be prompted to associate the GSD configuration file with a bitmap image of the slave device. Use the File Open dialog box to browse for the location of the image file to use.

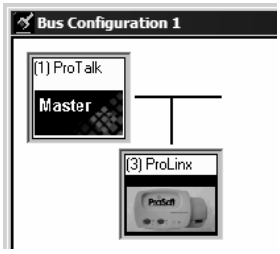
Note: This procedure does not automatically copy GSD configuration files from their original location to the GSD file path. In order to load GSD files automatically the next time you start ProSoft Configuration Builder, copy the files to the configuration file path in the Tip above.

The next step is to add the PROFIBUS Slaves to the network.

2.3.2 Configure the PROFIBUS Slaves

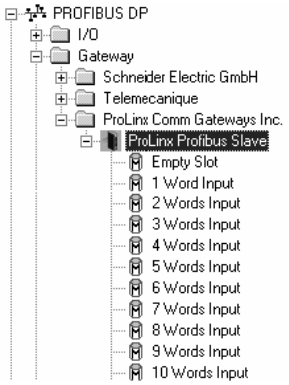
In this part of the procedure, you will add and configure the PROFIBUS slaves. In the following steps, you will add and configure a ProLinx PROFIBUS Slave module. The configuration information (.GSD file) for this module is provided on the PTQ-PDPMV1 Solutions CD-ROM.

1. In ProSoft Configuration Builder for PROFIBUS, click the plus sign [+] to expand the PROFIBUS DP tree.
2. Navigate to the folder containing the type of slave device to add, and then click the plus sign **[+]** to expand the folder.
3. Drag the Slave icon into the Bus Configuration window. The slave device appears in the Bus Configuration window as a network location to the Master.

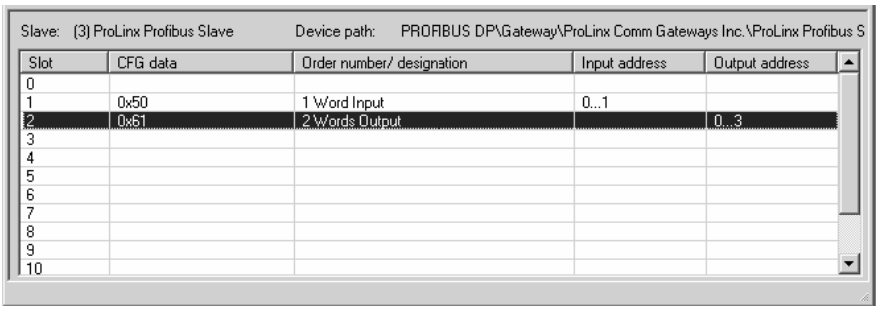


4. In the tree view, click the plus sign **[+]** to expand the slave device you added. This action opens a list of device configuration values. The following illustration shows the device configuration values for a ProLinx PROFIBUS Slave. The values for other devices may be different, so you should review

the specifications for the product you are installing in order to determine the correct values to use.

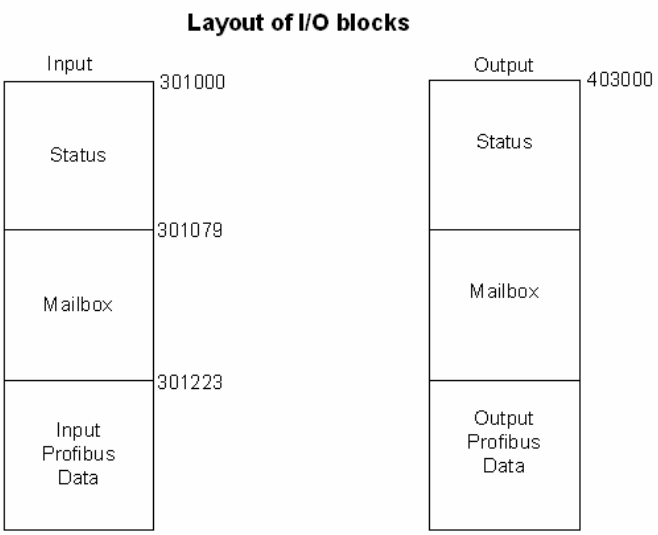


5 Drag the input and output parameters to the slot location grid below the Bus Configuration window. This view displays the configuration data, order number, and starting input and output addresses.



Important: The starting input and output addresses that you select here are actually byte offsets within the PROFIBUS Data area inside each Input and Output backplane block.

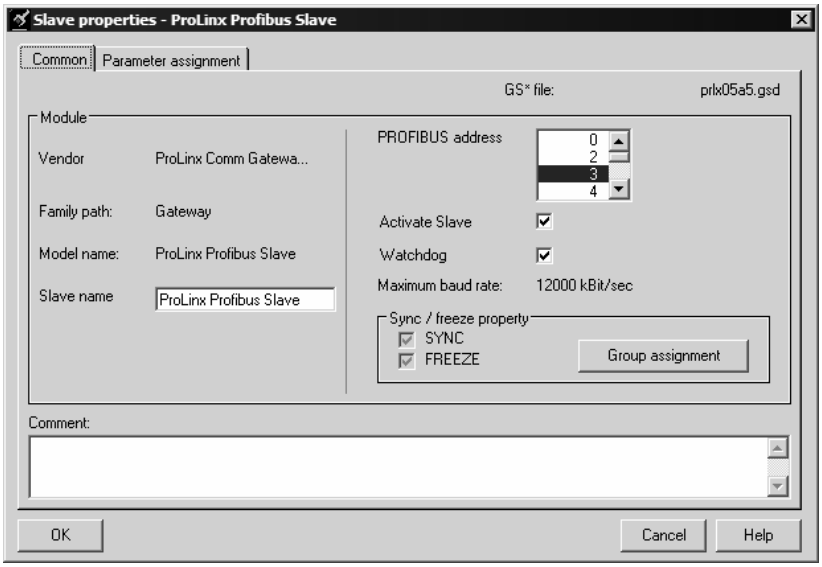
For example, for the sample configuration for the input block, where the Input Start Register Parameter = 1000:



The following table shows the actual Quantum address:

Input Address configured in PCB (Bytes)	Actual Quantum Input Register Address (Words)
0...1	301223
2...3	301224
4...5	301225
...	...

1 Double click the Slave icon to view the Slave Device properties.



In particular, note the following settings:

- Automatic PROFIBUS Address Assignment:
  - ProSoft Configuration Builder automatically assigns a PROFIBUS address to each new slave. The address assignment begins at address 3, and is incremented by 1 for each new slave added to the network. You can change the address in the Common tab of the Slave Properties dialog box.
  - Automatic Input/Output Address Assignment
    For each new slave added to the PROFIBUS network, ProSoft Configuration Builder automatically converts the input/output byte addresses to word input/output addresses for the State RAM in the processor.

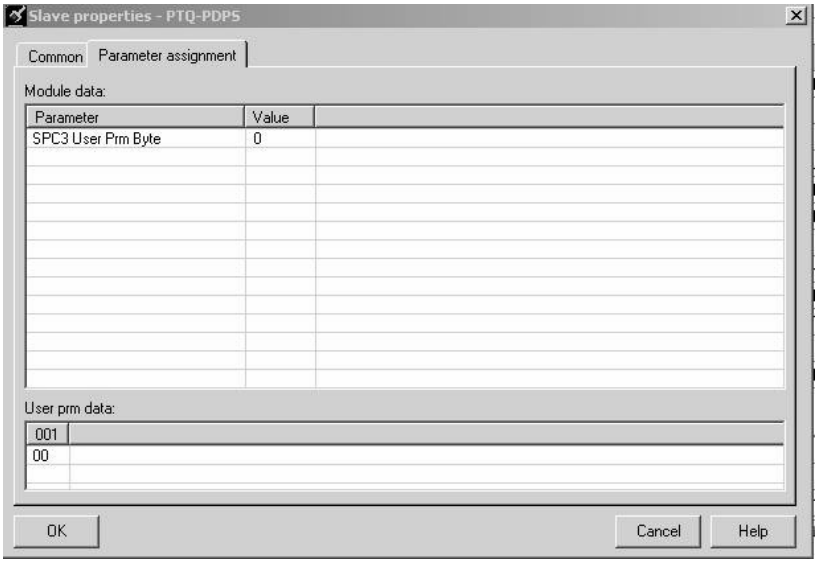
1 Repeat steps 2 through 6 for all slaves you intend to place on the network.

2 When you are finished adding slaves, open the Project menu and choose Exit to return to the Master Setup dialog box.

The next step is to export the memory maps and download the project to the module.

Set_Param (SAP61)

ProSoft PROFIBUS Slave (PDPS) devices have a configurable parameter for SPC3 User Prm Byte. The following illustration shows the value of this parameter in ProSoft Configuration Builder for PROFIBUS, the configuration tool for ProSoft PROFIBUS Master devices.



Parameter Data Structure

SPC3 evaluates the first seven data bytes (without user prm data), or the first eight data bytes (with user prm data). The first seven bytes are specified according to the standard. The eighth byte is used for SPC3-specific communications. The additional bytes are available to the application.

Byte	Bit Position								Designation
	7	6	5	4	3	2	1	0	
0	Lock Reg	Unio Req	Sync Req	Free Req	WD on	Res	Res	Res	Station status
1									WD_Fact_1
2									WD_Fact_2
3									MinTSDR
4									Ident_Number_High
5									Ident_Number_Low
6									Group_Ident
7									Spec_User_Prm_Byte
8 to 243									User_Prm_Data

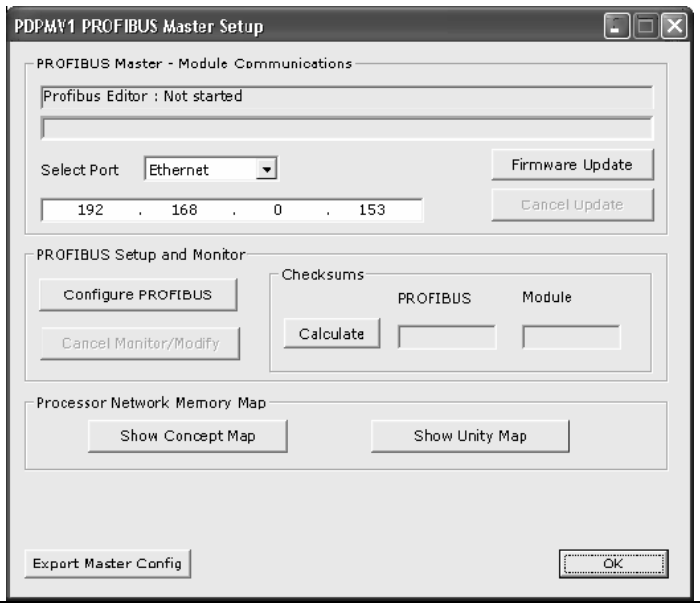
Byte 7	Spec_User_Prm_Byte		
Bit	Name	Significance	Default State
0	Dis_Startbit	The start bit monitoring in the receiver is switched off with this bit	Dis_Startbit = 1, That is, start bit monitoring is switched off.
1	Dis_Stopbit	Stop bit monitoring in the receiver is switched off with this bit	Dis_Stopbit = 0 That is, stop bit monitoring is not switched off.
2	WD_Base	This bit specifies the time base used to clock the watchdog. WD_Base = 0: time base 10 ms WD_Base = 1: time base 1 ms	WD_Base = 0 That is, the time base is 10 ms.
3 to 4	Res	To be parameterized with 0	0
5	Publisher_Enable	DXB-publisher-functionality of the SPC3 is activated with this bit	Publisher_Enable = 0, DXB-request-telegrams are ignored; Publisher_Enable = 1, DXB-request-telegramme are processed
6 to 7	Res	To be parameterized with 0	0

2.3.3 Export the Processor Memory Map

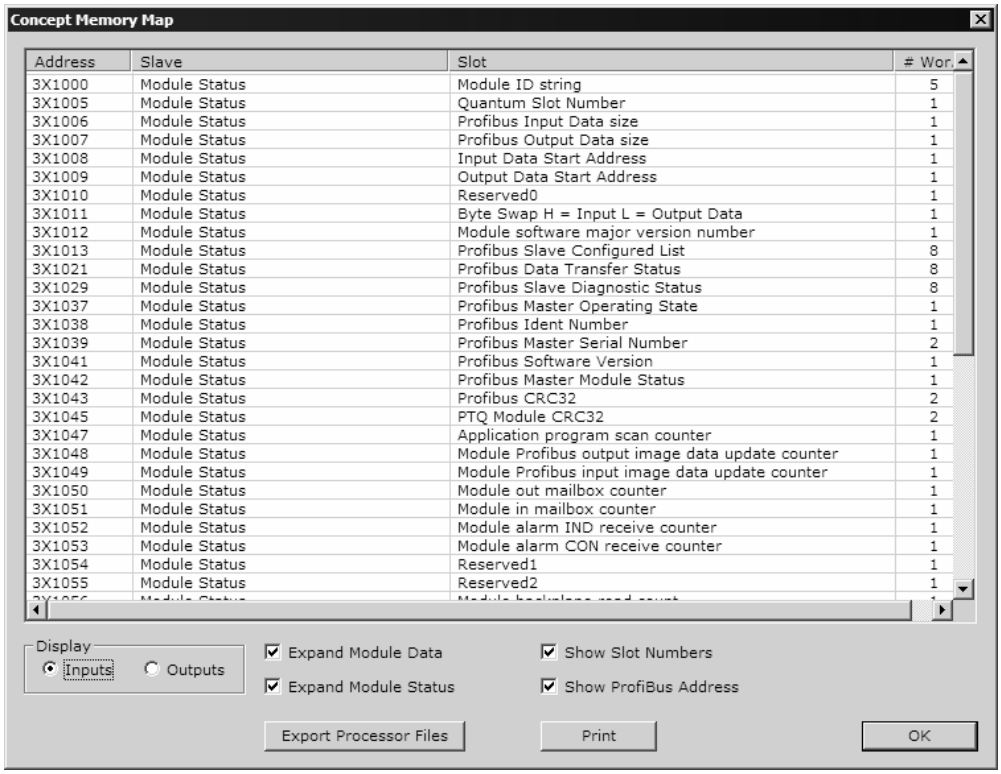
The import file (PTQ_PDPMV1.XSY for Unity, or PTQ_PDPMV1.DTY for Concept) that you create in this step uses the information in the Processor Memory Map to build the derived data tags for the slave devices on your PROFIBUS network. These tags allow the program running on the processor to access data within the module.

➢ ***To export the processor memory map:***

1 In the Master Setup dialog box, click **Show Concept Map** (for processors configured with Concept software) or click **Show Unity Map** (for processors configured with ProSoft Configuration Builder software).



2 This action opens the Memory Map dialog box.


Concept Memory Map
Address
Slave
Slot
Wor.
3X1000 Module Status Module ID string 5
3X1005 Module Status Quantum Slot Number 1
3X1006 Module Status Profibus Input Data size 1
3X1007 Module Status Profibus Output Data size 1
3X1008 Module Status Input Data Start Address 1
3X1009 Module Status Output Data Start Address 1
3X1010 Module Status Reserved0 1
3X1011 Module Status Byte Swap H = Input L = Output Data 1
3X1012 Module Status Module software major version number 1
3X1013 Module Status Profibus Slave Configured List 8
3X1021 Module Status Profibus Data Transfer Status 8
3X1029 Module Status Profibus Slave Diagnostic Status 8
3X1037 Module Status Profibus Master Operating State 1
3X1038 Module Status Profibus Ident Number 1
3X1039 Module Status Profibus Master Serial Number 2
3X1041 Module Status Profibus Software Version 1
3X1042 Module Status Profibus Master Module Status 1
3X1043 Module Status Profibus CRC32 2
3X1045 Module Status PTQ Module CRC32 2
3X1047 Module Status Application program scan counter 1
3X1048 Module Status Module Profibus output image data update counter 1
3X1049 Module Status Module Profibus input image data update counter 1
3X1050 Module Status Module out mailbox counter 1
3X1051 Module Status Module in mailbox counter 1
3X1052 Module Status Module alarm IND receive counter 1
3X1053 Module Status Module alarm CON receive counter 1
3X1054 Module Status Reserved1 1
3X1055 Module Status Reserved2 1
Display
Inputs
Outputs
Expand Module Data
Expand Module Status
Show Slot Numbers
Show ProfiBus Address
Export Processor Files
Print
OK

3 On the Memory Map dialog box, click **Export Processor Files**.

Note: For Unity Map, PCB will export the XSY file and XFM files in the same directory if mailbox parameter is chosen. The filenames will match the module name you chose in PCB.

For Concept Map, PCB will export .dty file, .txt file and .asc files if mailbox parameter is chosen.

4 Name the file and choose a location on your hard drive. The recommended location is your "My Documents" folder, and then click **Save.**

5 Click **Print** to print the input and output maps for reference.

When you import this memory map file into the processor configuration, it simplifies the task of establishing communications between the module and the processor. You will have to establish backplane communications using either Concept or Unity XL Pro software.

After you download the configuration to the PTQ-PDPMV1 module, save the .dty and .xsy files to a location on your hard drive (a folder below C:\PCBExportFiles), where you will import them into the processor during the processor configuration steps. These project files greatly reduce the amount of time it would otherwise take to perform the necessary configuration tasks.

Refer to ***Configuring the Processor with Unity PRO XL*** (page 39) and ***Configuring the Processor with Concept*** (page 73) for detailed instructions on how to configure the processor.

Note: The recommended location for the files is the "My Documents" folder on your PC. The configuration tool for the processor will use this folder by default.

The next step is to calculate the checksums for the PROFIBUS configuration and PTQ Module configuration.

Calculate Checksums

The checksum (CRC) values are calculated from the PROFIBUS configuration data, and compare the contents of the configuration file in the module with the value reported by the processor. The checksum (CRC) value allows the processor to verify that the configuration file is valid, and has not changed since the last time the configuration file was imported to the processor. Any change to the contents of the configuration file in either location changes the unique numeric (CRC) value for the file.

If the checksum values do not match, the Master stops and indicates a configuration error, and the CFG light illuminates on the module.

1. On the PTQ-PDPMV1 PROFIBUS Master Setup dialog box, click the Calculate Checksums button.
2. Make a note of the checksum values so that you can enter them later if prompted.
3. To insert the checksum values in UnityPro XL, refer to ***Updating Checksum Values - UnityPro XL*** (page 44).
   To insert the checksum values in Concept, refer to ***Configuration Validation & SETCRC Function Block*** (page 94).

The next step is to download the project to the PTQ module.

2.3.4 Download the Project to the Module

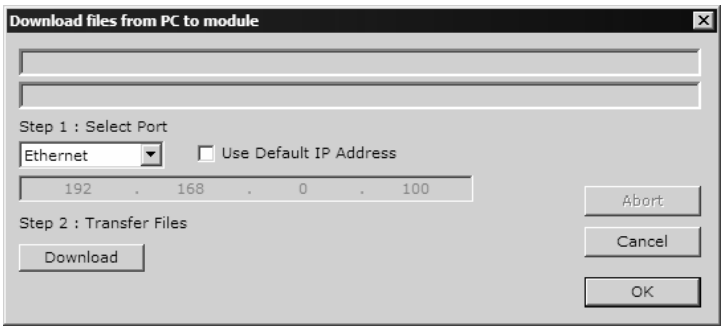
In order for the module to use the PROFIBUS network settings you configured, you must download (copy) the updated Project file from your PC to the module.

Note: The processor (Quantum) must be in "Stop" mode before you download the file to the module. Use the processor's configuration tool or the softkeys on the processor to stop the processor.

➢ ***To Download the Project File***

1. In the tree view in ProSoft Configuration Builder, click once to select the PTQ-PDPMV1 module.

2 Open the **Project menu,** and then choose **Module** → **Download**. This action opens the dialog box shown below.



3 Choose Ethernet from the dropdown list, and then click the Download button. When the download is complete, a dialog box will prompt you to place the processor back into "Run" mode.

- **Note:** If you have not yet downloaded the Ethernet Configuration (WATTCP.CFG) file, which contains the customized IP address settings for the module, you have the option on this dialog box to connect using the module's default IP address (192.168.0.100).
- **HSBY Note:** For HSBY Ethernet downloading (Ethernet recommended), both HSBY modules must be connected to allow PCB to download to both modules. PCB will download to the first Master, and will then prompt you to download the project to the second module Master.

The module will perform a platform check to read and load its new settings. When the platform check is complete, the status bar in ProSoft Configuration Builder will be updated with the message *"Module Running".*

2.3.5 Back Up the Project

In this step, you will create a backup copy of your project and configuration files. The backup procedure saves your data for reuse on another machine, or allows you to restore your data in the event of a system failure.

➢ ***To save your project and configuration files:***

1 In ProSoft Configuration Builder tree view, click [+] to expand the PTQ-PDPMV1 tree, and then double-click the PROFIBUS DP icon. This action opens the PROFIBUS Master Setup dialog box.

2 In the PROFIBUS Master Setup dialog box, click the Export Master Config button. This action saves the PROFIBUS network configuration for your module in an XML file. The recommended location for this file is your "My Documents" folder.

Tip: You can use the XML file created by ProSoft Configuration Builder in this step to simplify the task of configuring additional PROFIBUS network modules. Because its saves the entire network configuration, you can add modules quickly by modifying only the items that are unique for each device, typically the slot number and I/O addresses. To use this saved configuration, open Windows Explorer, navigate to the folder where you saved the Master Configuration XML file, and then drag the file onto the new PROFIBUS DP icon in the ProSoft Configuration Builder tree view.

3 **Unity Pro XL Users:** From the PROFIBUS Master Setup Screen, click the **Show Unity Map** button, then click the **Export Processor Files** button. This action exports the xfm file (created only if the Mailbox parameter is set to Yes) and xsy file. The recommended location for these files is your "My Documents" folder.
Concept Users: From the PROFIBUS Master Setup Screen, click the **Show Concept map** button, and then click **Export Processor Files.** This action exports the DTY and related files. The recommended location for these files is your "My Documents" folder.
4 From PROFIBUS Master Setup, click the **Show Concept map** button. Then choose **Export Processor Files** to export the DTY, TXT and other related files if the Mailbox parameter is set to Yes. The recommended location is your "My Documents" Folder.
5 Click OK to close the PROFIBUS Master Setup dialog box.
6 In the ProSoft Configuration Builder, open the File menu, and then choose Save As.
7 Name the project file, and click Save. The recommended location for this file is your "My Documents" folder.

Note: All PCB project files and module related files are automatically saved to C:\PCBExportfFiles.

A complete backup consists of the Project and Master Configuration files, plus the GSD configuration files. The default location for the GSD files is C:\Documents and Settings\All Users\Application Data\ProSoft\GSD (Windows XP / 2000) or C:\My Documents\. To move a project to a different PC, copy the .PPF, .XML and .GSD files to the same directory structure on the new machine that they occupied on the old one.

The above method defines a manual approach in creating Quantum processor I/O and Function Block import files. The PCB will also automatically create these files when the PCB project is saved or closed (if the project is not saved then PCB will not export the files).

You can also generate these files manually from PCB. To create the files:

1 Open the Project menu, and select Project / Export Files.
2 If you are prompted to overwrite files, click Yes.
File Locations The folder structure implemented for each PCB project (ppf) is as follows:

{rootdrive}\PCBExportFiles\'ppf name'\'Project Name'\'Location Name'\'Module Name'\

The following files will be created in each folder created by PCB

- {rootdrive}\PCBExportFiles\'ppf name'\
  - Project ppf file (.ppf)
- {rootdrive}\PCBExportFiles\'ppf name'\'Project Name'\'Location Name'\
  - (Concept folder created only for PTQ-PDPMV1 modules)
  - \Concept\.dty, .asc files
- {rootdrive}\PCBExportFiles\'ppf name'\'Project Name'\'Location Name'\'Module Name'\
  - PROFIBUS xml file (modulename{ModuleName}.xml) PTQ cfg file (.cfg)
  - (Unity folder created only for PTQ-PDPMV1 modules)
  - \Unity\Unity xml files (.xsy, .xfm)
- (gsd folder created for all PDPMV1 modules)
- \gsd\GSD files used for module (.gsd)
- (Concept folder created only for PTQ-PDPMV1 modules)
- \Concept\txt files for variables

2.4 Configure the Processor

If you have followed the previous steps in order, your PTQ module is now configured with the settings for your PROFIBUS Master and Slaves. The final task is to import this information into the processor. This task allows the processor to communicate with the PTQ module and its slave devices over the backplane. The following topics will describe the different procedures for Unity and Concept platforms.

IMPORTANT NOTE: The following four steps are required in order to get the system will be up and running without any complications.

1. Download configuration to module from PCB
2. Export files (XFM and XSY) from PCB
3. Import .XFM file that was exported in step 2.
4. Import .XSY file that was exported in Step 2.

3 Configuring the Processor with Unity Pro XL

In This Chapter

- Import the Functional Module 39
- Function Blocks Operation Overview 54
- Derived Function Blocks Overview 57
- Using Mailbox Function Blocks 60
- Mailbox Overview 64

3.1 Import the Functional Module

To simplify the task of programming the processor when communicating with the PTQ-PDPMV1 module, ProSoft Technology has created a Unity Pro XL Functional Module type (XFM).

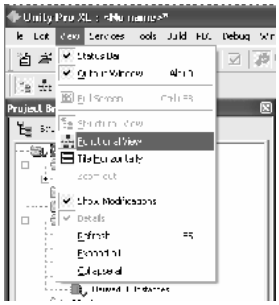
Warning: The Functional Module is intended for new installations of PTQ-PDPMV1. If you have an existing installation, the following procedure will overwrite your settings, and may cause loss of functionality. DO NOT overwrite a working application until you have thoroughly reviewed the following topics.

The Functional Module provides easy access to PROFIBUS slaves cyclic data and the PTQ module's input/output status data. Specific mailbox commands are provided to perform DPV0/V1 acyclic functions such as "Get Live List", "Get Slave Diagnostics", and perform Freeze and Sync commands. The Functional Module exchange file name matches the module name you defined in PCB, with the extension .XFM. This file is created by PCB when you export the processor file from the ***Show Unity Map dialog box*** (page 36).

➢ *To import the Functional Module:*

Use the project you created in Unity Pro XL, and perform all of the following steps.

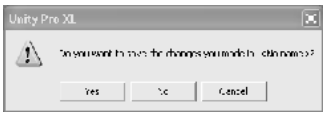
1 Open the View menu, and then choose Functional View.



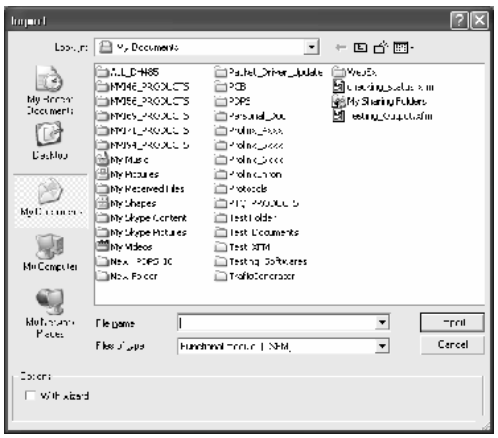
This action populates the Project Browser with a Functional Station icon, as shown in the following illustration.

2 Select Functional Station, and then click the right mouse button to open a shortcut menu. On the shortcut menu, choose Import.

Click No to dismiss the confirmation dialog box.



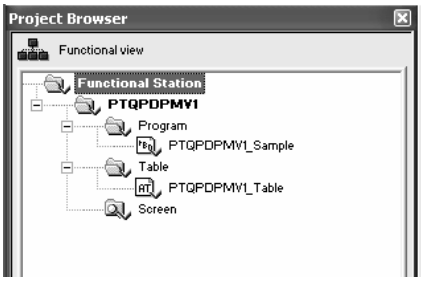
In the Import dialog box, choose Functional module (*.XFM) in the Files of Type dropdown list, and then select the XFM file to import. The XFM file name matches the module name you defined in PCB and exported in step 3 of ***Back Up the Project*** (page 36).



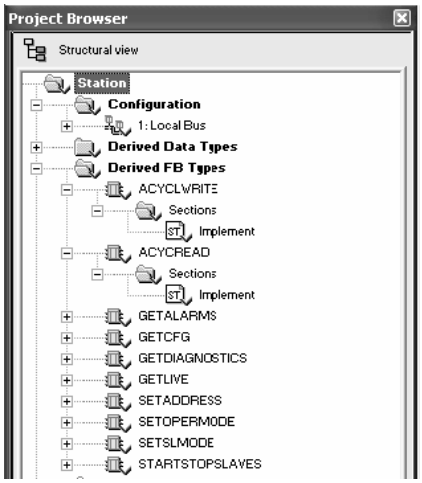
Click Import to import the file.

Note: Use the XFM file created by PCB. The XFM file created by PCB is preferred, because it contains the I/O map representing your PROFIBUS network and contains the same variable names. This file will be created only if Mailbox messaging Parameter is set to YES

Notice that the Project Browser is now populated with the Functional Module.



3 To view the DFBs, data types and variables associated with the Functional Module, open the View menu and choose Structural View. Notice that all function blocks have been defined using the ST type language.



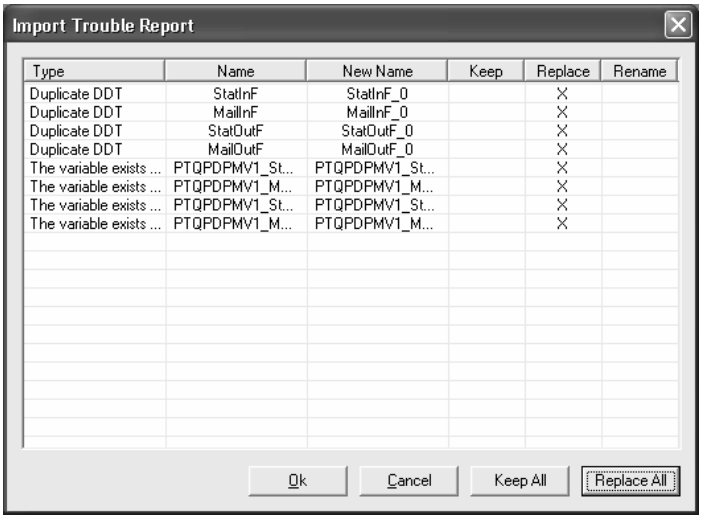
➢ *To import the Variables*

Import the PROFIBUS I/O table, contained in the xsy file user created when exported the memory map from ProSoft Configuration Builder (PCB) in **Back Up the Project**. This file contains all the cyclic input and output variables configured by the PCB master configuration software. This file also includes Module status data and may also include slave diagnostics data and mailbox data if these parameters were chosen.

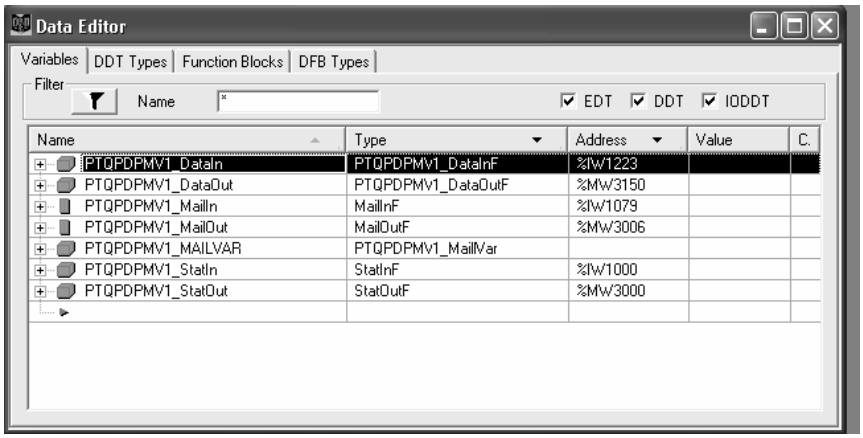
1 In the Project Browser, select Variables & FB Instances, and click the right mouse button to open a shortcut menu. On the shortcut menu, choose Import.

2 In the Files of Type: dropdown list, choose Data Exchange File (*.XSY). Select the .xsy file created in the **Back Up the Project**, and then click Import.

3 In the Import Trouble Report window, click Replace All, then click OK



At this point the input and output variables have been imported to the application.

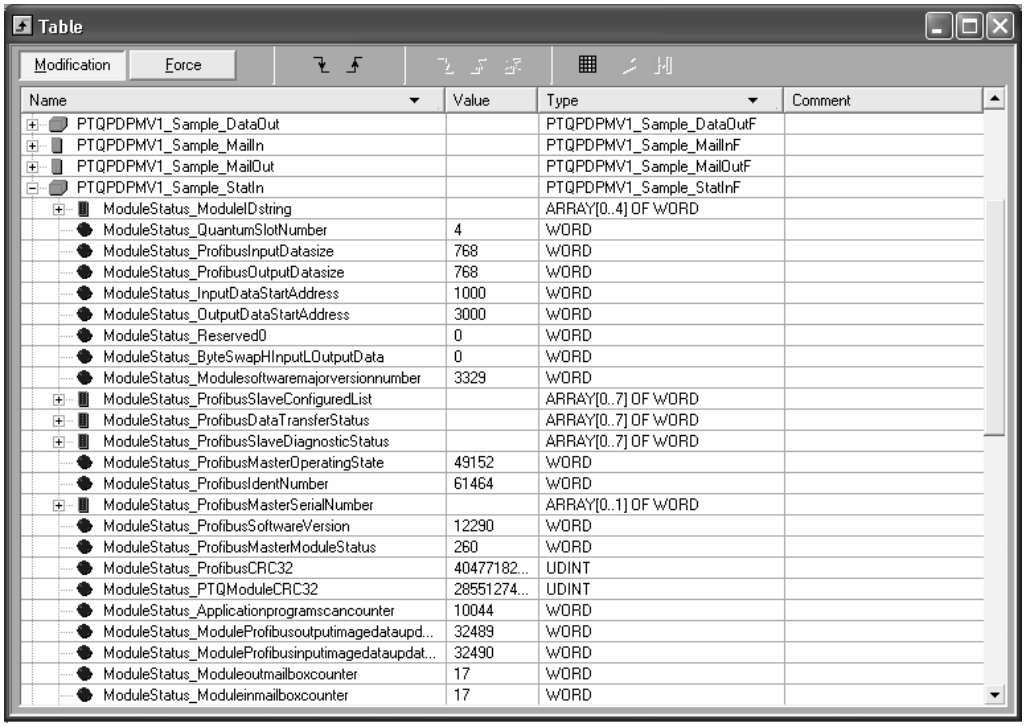


➢ *To Modify an Animation Table*

Note: An Animation table is required to send and receive mailbox messages, monitor State Ram status and read/write IO data.

An Animation Table is provided with the XFM file, but certain data variables must be added to monitor the status or the health of the module.

Double-click the animation table and under the name column select the {ModuleName}_StatIn variables. Under the <inputs> folder, select the Modification Tab. You should see the module status counters update.



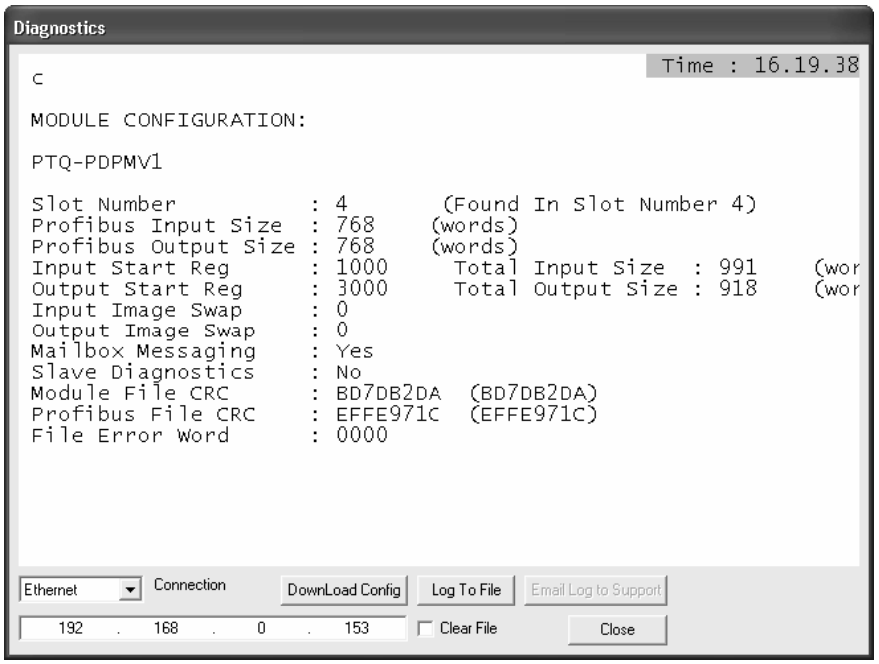
3.1.1 Updating Checksum Values - UnityPro XL

The PTQ-PDPMV1 module is almost ready, and the CRC values for the PROFIBUS configuration should match between the module and the processor.

➢ ***To confirm that both CRCs match:***

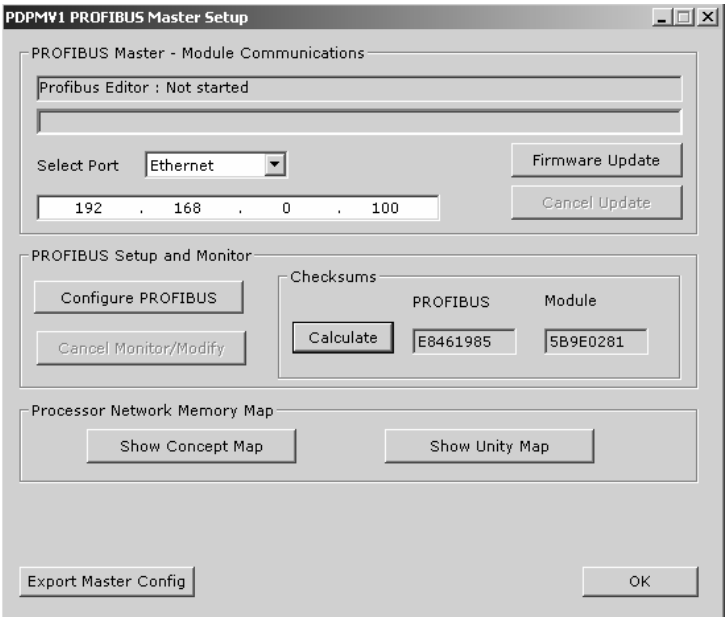
1. From PCB, select the module icon, and then click the right mouse button to open a Shortcut menu.
2. On the shortcut menu, choose Diagnostics. Wait for ProSoft Configuration Builder to go online with the module through the Serial or Ethernet port.
3. When the module is online, press [?] to display the main menu.
4. On the main menu, press [C] to view the module configuration. The following illustration shows example CRC values for the Module File and the PROFIBUS File.

Note: Because the CRC values are calculated for your unique configuration, the values on your screen will not be the same as the ones in the following illustration.


Diagnostics
Time : 16.19.38
C
MODULE CONFIGURATION:
PTQ-PDPMV1
Slot Number : 4 (Found In Slot Number 4)
Profibus Input Size : 768 (words)
Profibus Output Size : 768 (words)
Input Start Reg : 1000 Total Input Size : 991 (wor
Output Start Reg : 3000 Total Output Size : 918 (wor
Input Image Swap : 0
Output Image Swap : 0
Mailbox Messaging : Yes
Slave Diagnostics : No
Module File CRC : BD7DB2DA (BD7DB2DA)
Profibus File CRC : EFFE971C (EFFE971C)
File Error Word : 0000
Ethernet
Connection
DownLoad Config
Log To File
Email Log to Support
192 . 168 . 0 . 153
Clear File
Close

➢ *To calculate checksums*

1 On the PTQ-PDPMV1 PROFIBUS Master Setup dialog box, click the Calculate Checksums button.

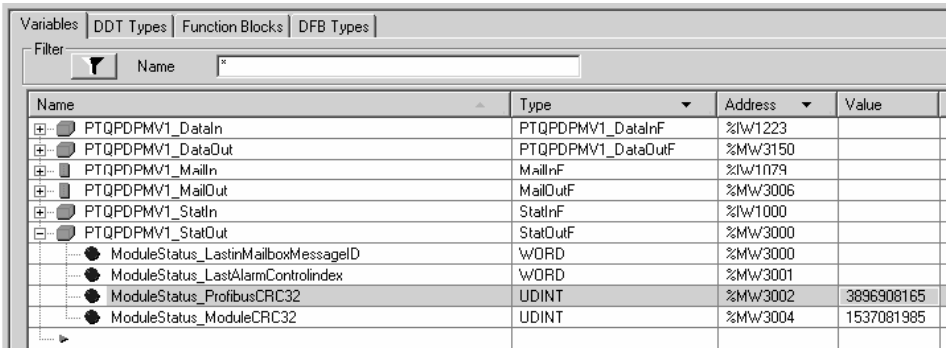


Notice the NEW checksums for the Module and PROFIBUS appear.

Note: The module checksum will change when parameters such as 3X or 4X starting address are changed and the PROFIBUS checksum will change if a network parameter is changed.

2 Copy the PROFIBUS checksum by highlighting the text and right click to Copy.

3 Go to the Unity Pro Variables Tab and select the "_StatOut" variable and expand the structure to expose the "ModuleStatus_ProfibusCRC32" element. Under the Value column area, paste the copied checksum.



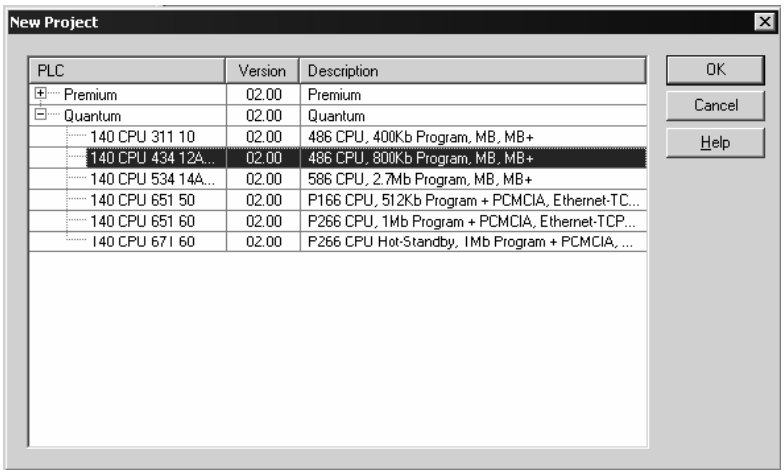
4 Repeat steps 2 and 3 above for the Module checksum value. Paste the value into "ModuleStatus_ModuleCRC32" variable.

5 Download the new values in the program to the processor.

3.1.2 Setup General Unity Pro XL Project Settings

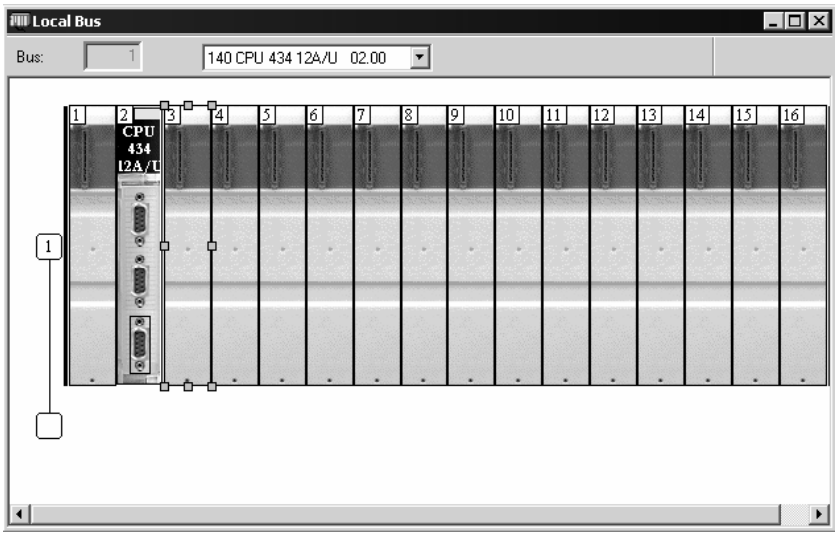
To setup general Unity project settings, follow these steps.

1 Start Unity Pro XL. Open the File menu, and then select New. This action opens the New Project dialog box.

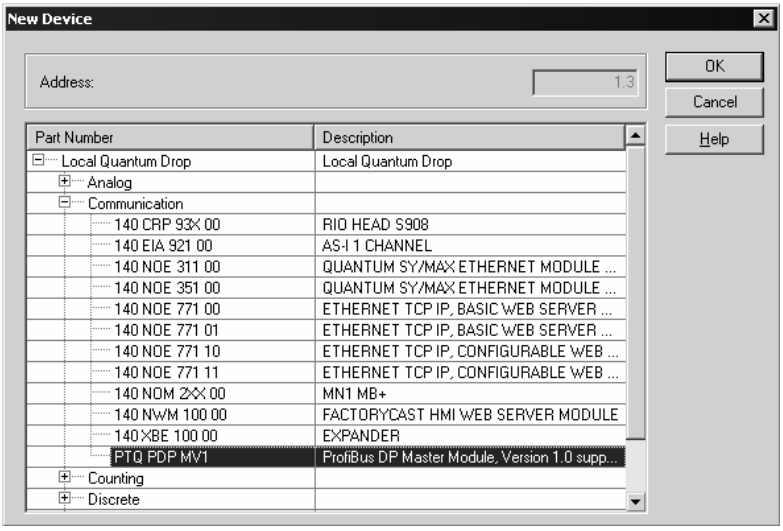


2 The dialog box shows a list of processors that it can configure. Choose the processor you are configuring from the list, and then click OK to open the Project Browser.

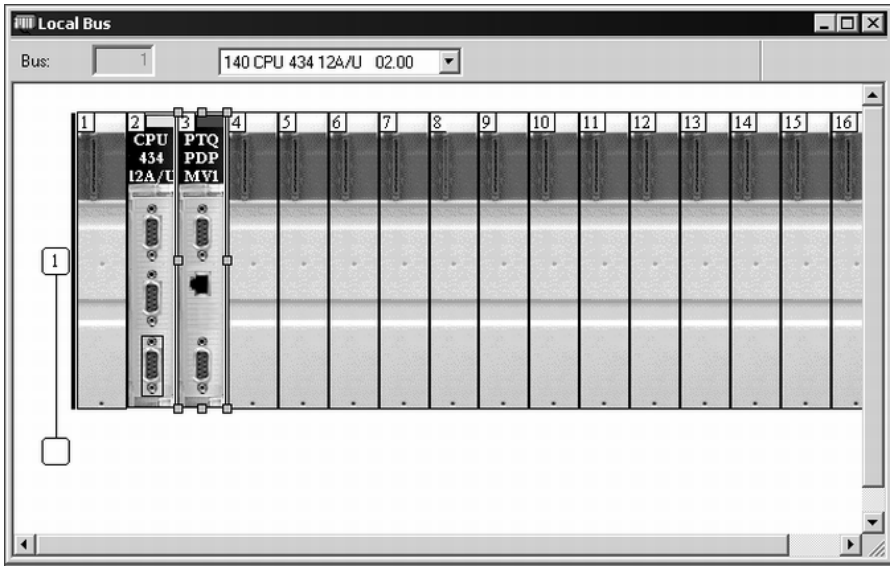
3 In the Project Browser tree view, double-click **Local Bus** to open the *Local Bus* window. Notice that the image in the window shows the processor in the second position in the rack (the first position is for the power supply, which you will add later. In the following steps, you will add an image of the PTQ module to the rack, in the same position where you physically installed the module.



4 To add devices to the rack, double-click the location in the rack where the device is installed. This action opens the New Device dialog box.



5 Click the **[+]** sign next to Communication to open the list of Communication devices. Select PTQ PDPMV1 from the list, and then click OK. This action adds the module to the Local Bus image.



6 Repeat the previous two steps to add other devices, such as power supplies, to the rack.

7 When you have finished adding devices, open the File menu and choose Save. This action saves the project to the hard drive on your PC.

3.1.3 Configure the Memory Size for the Processor

Part of the processor configuration process allocates memory to use in the processor to store input and output data from the module. For installations where the processor communicates with only one module, the default memory settings will work without further configuration. The following steps will help you determine the correct memory addresses to assign for more complex installations.

The processor memory maps that you configured in ProSoft Configuration Builder are exported from ProSoft Configuration Builder, and imported into the Unity Pro XL project. These values are calculated from the starting memory address in the processor's State RAM for the module's input and output data images. Refer to ***Configure the Module*** (page 14) for more information on configuring memory addresses in ProSoft Configuration Builder.

Depending on the complexity of your installation, for example when you are deploying the PTQ-PDPMV1 module in an existing system, you should view the memory configuration for the processor in ProSoft Configuration Builder before you begin to configure memory addresses in Unity Pro XL.

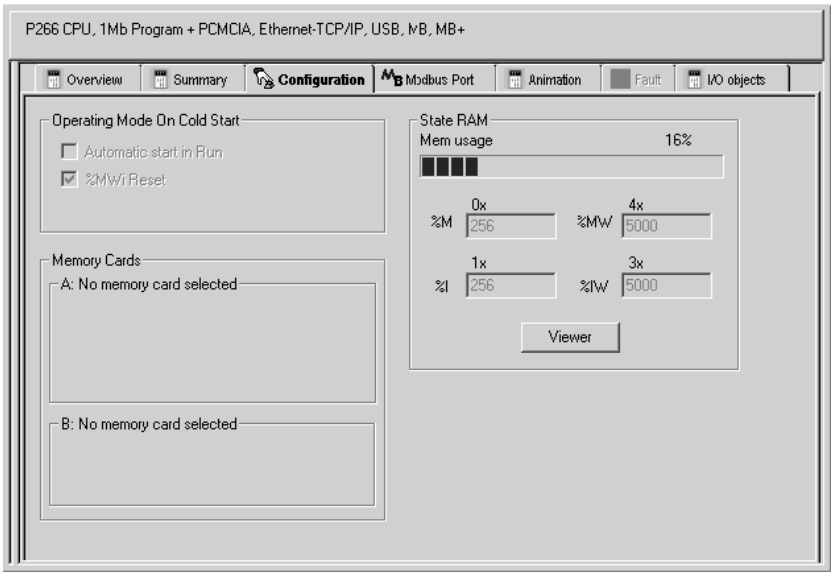
Some points to keep in mind are:

- As the programmer, you must be aware of the memory spaces that are available when deploying in an existing system, and assign values to the PTQ accordingly.
- Data registers must exceed starting registers. This is in the memory map page that you printed.
- You must assign the PTQ module to a block of processor memory that is not being used by any other device.
- You can use this simple formula to find a block of memory to use: If the module consumes 224 words of status data on input, and we know that it can take up to 768 words of I/O data, the total requirement is 992 words. The module will take a maximum of this %IW value. For convenience, round the number up to 1000 as the amount of memory to assign. A value of 5000 for %MW and %IW is a safe starting point.
- It is not possible to determine if the memory values are correct before building the project. If the build throws an error about memory addresses, go back to ProSoft Configuration Builder and change the input and output properties for the module, then re-import the memory map and try again.

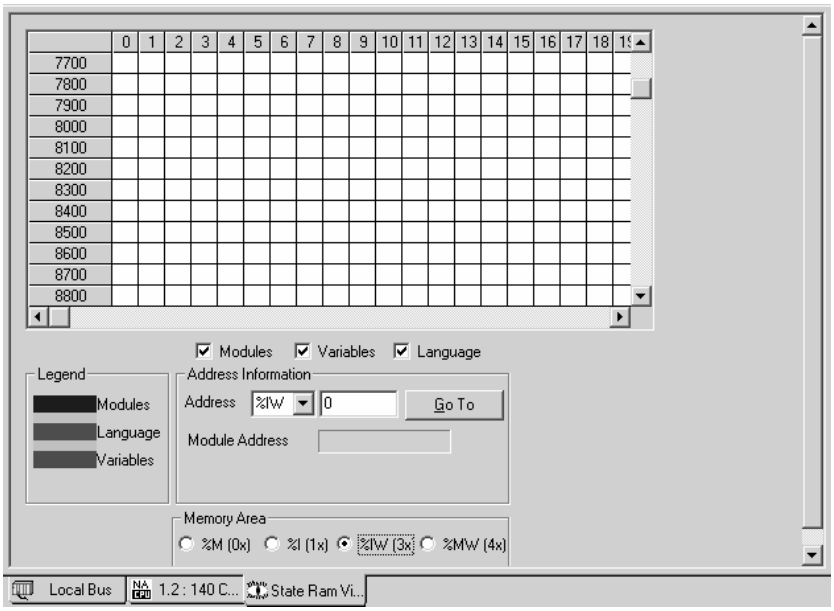
➢ ***To view memory usage in the processor:***

1 Start Unity Pro XL.
2 In the Project Browser, expand the Configuration tree, and then double-click the Local Bus object.
3 In the Local Bus window, double-click the processor. This action opens a tabbed window with information about the processor.

4 Click the Configuration tab. This tab provides an overview of the processor's memory configuration.



5 To view detailed information about the processor's memory configuration, click Viewer. The viewer offers tools to view the types of data stored at specific addresses in the processor. Make note of memory areas that are already allocated, and select an area of contiguous memory that be allocated to the PTQ module.



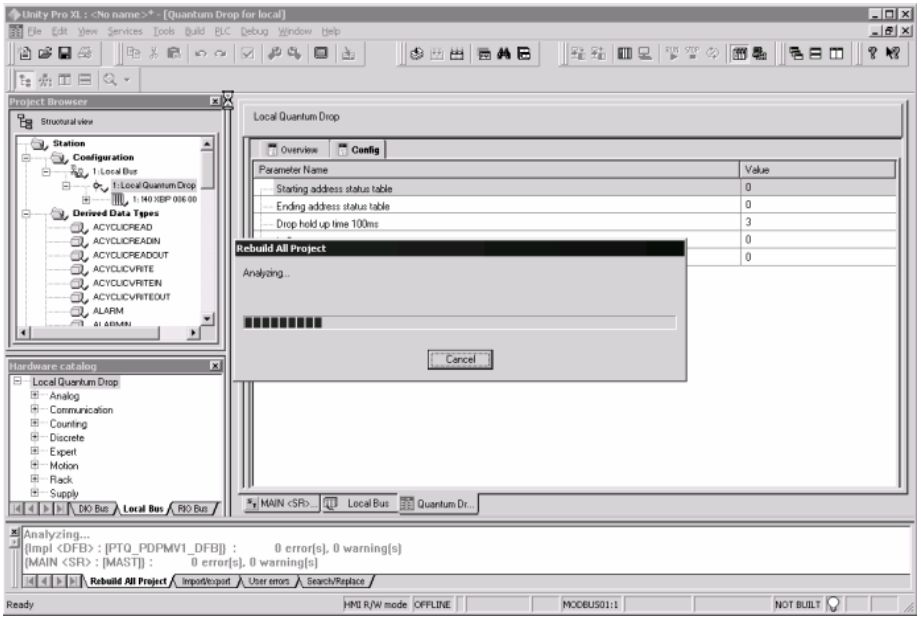
3.1.4 Build the Project

Whenever you update the configuration of your PTQ module, the PROFIBUS network, or the processor, you must import the changed configuration from the module, and then build (compile) the project before downloading it to the processor.

Note: The following steps show you how to build the project in Unity Pro XL. This is not intended to provide detailed information on using Unity Pro XL, or debugging your programs. Refer to the documentation for your processor and for Unity Pro XL for specialized information.

➢ ***To build (compile) the project:***

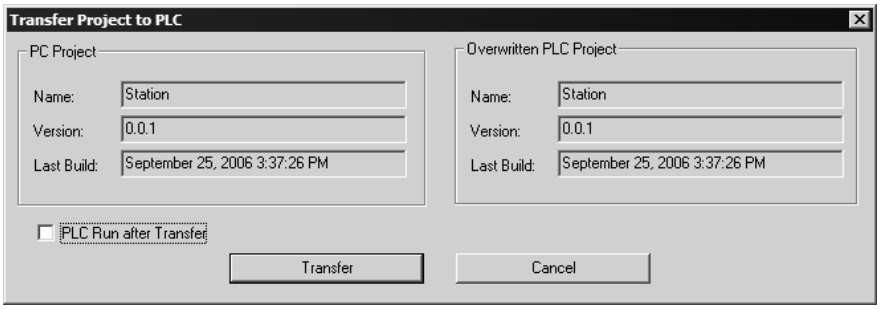
1. Review the elements of the project in the Project Browser.
2. When you are satisfied that you are ready to download the project, open the Build menu, and then choose Rebuild All Project. This action builds (compiles) the project into a form that the processor can use to execute the instructions in the project file. This task may take several minutes, depending on the complexity of the project and the resources available on your PC.
3. As the project is built, Unity Pro XL reports its process in a Progress dialog box, with details appearing in a pane at the bottom of the window. If you are using the sample project, the project should build without errors. The following illustration shows the build process under way.



After the build process is completed successfully, the next step is to download (copy) the compiled project to the processor.

3.1.5 Download the Project to the Processor

1 Open the PLC menu and then choose Connect. This action opens a connection between the Unity Pro XL software and the processor, using the address and media type settings you configured in the previous step.
2 On the PLC menu, choose Transfer Project to PLC. This action opens the Transfer Project to PLC dialog box. If you would like the PLC to go to "Run" mode immediately after the transfer is complete, select (check) the PLC Run after Transfer check box.



3 Click the Transfer button to download the project to the processor. As the project is transferred, Unity Pro XL reports its process in a Progress dialog box, with details appearing in a pane at the bottom of the window.

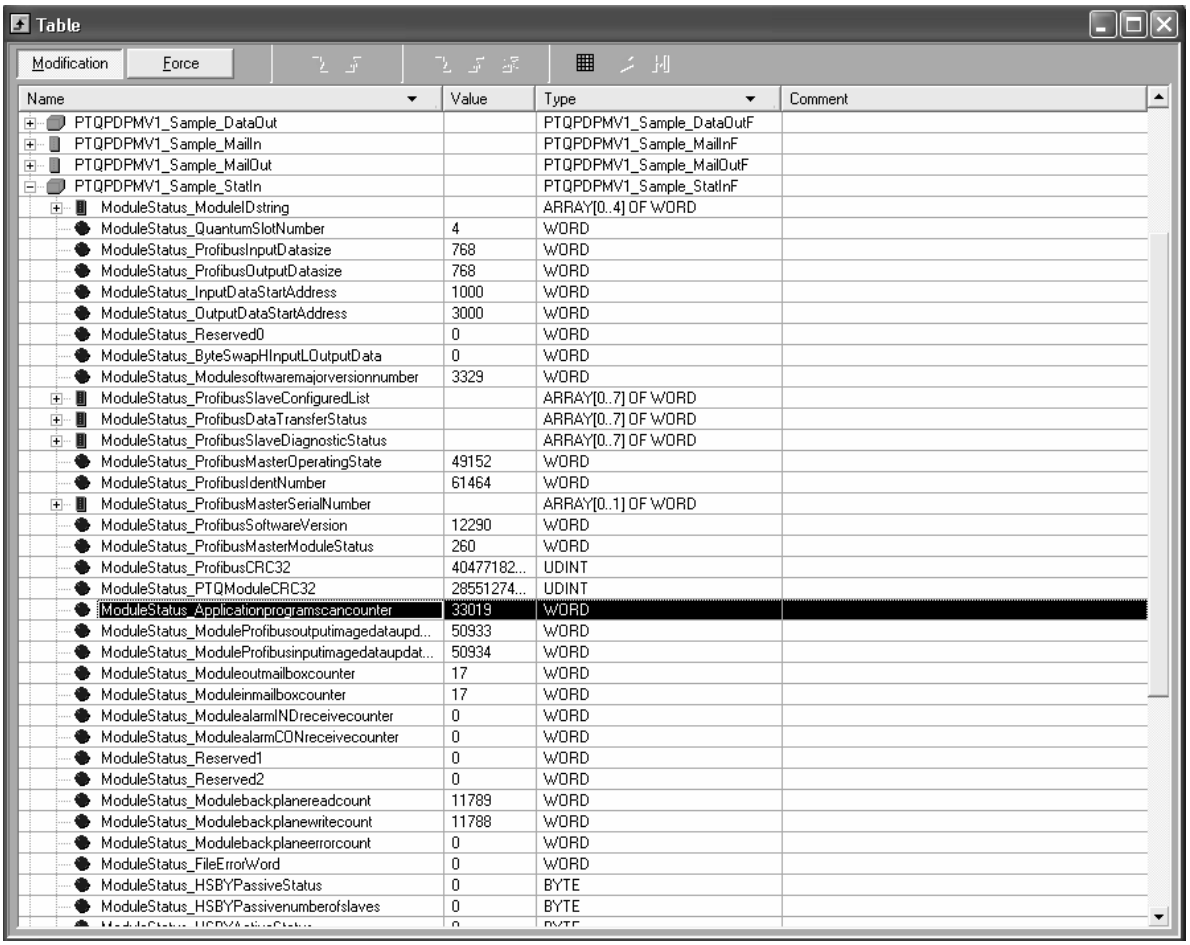
When the transfer is complete, place the processor in Run mode.

3.1.6 Verify Communication between the Processor and the Module

In this step, you will verify that the processor and the PTQ module are communicating with each other over the backplane. The sample project includes an animation table called MailBox Commands. When the processor and the PTQ module are communicating, the values in this animation table are updated in real time.

➢ To verify communication between the processor and the module:

1 Place the processor in Run mode, if you have not already done so.
2 In the Unity Pro XL project browser pane, click **[+]** to open the Animation Tables tree, and then double-click Main Table.
3 In the Main:Table, you will see all mailboxes, including Get Live List, Get Diagnostics, and so on.
4 You must include **{ModuleName}_StatIn, {ModuleName}_MailIn** and **{ModuleName}_DataIn,** using same procedure for the Output **{ModuleName}_StatOut, {ModuleName}_MailOut** and **{ModuleName}_DataOut.**
5 Scroll within **{ModuleName}_StatIn**. Notice that when the processor and the PTQ module are communicating successfully, the numbers in the Value column for items such as ModuleStatus_Applicationprogramscancounter are continuously updated.

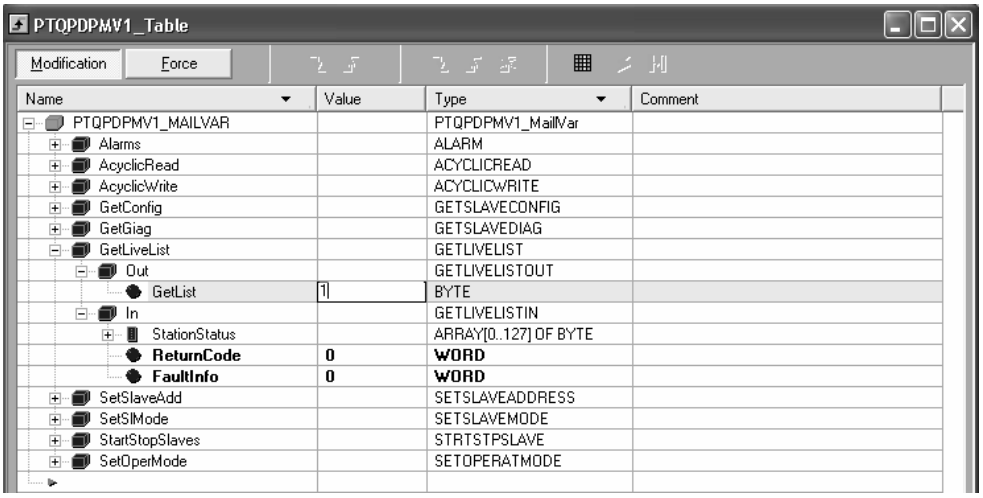


➢ *To Test the Unity Interface*

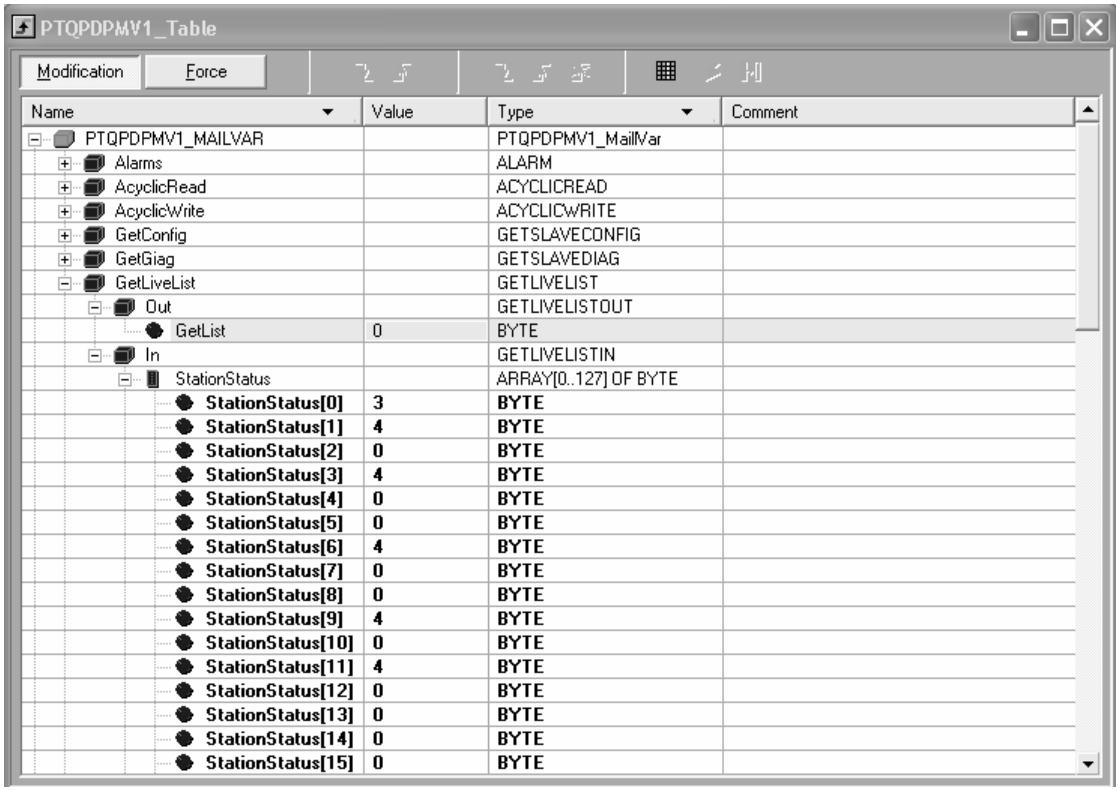
The following steps show how to use the mailbox message GetLiveList.

Note: Make sure the Unity program is connected and the processor is running

1 From the Table (public folder), select PTQPDPMV1_MAILVAR.GetLiveList.Out.GetList and set it to 1:



2 The GetLiveList response will be automatically copied at GetLiveList.In.StationStatus array. The following illustration shows an example where slave address 3 is connected to the master (address 1). The GetList bit is automatically cleared. Refer to ***Mailbox Messaging*** (page 112) for specific help on the mailbox commands and response values.

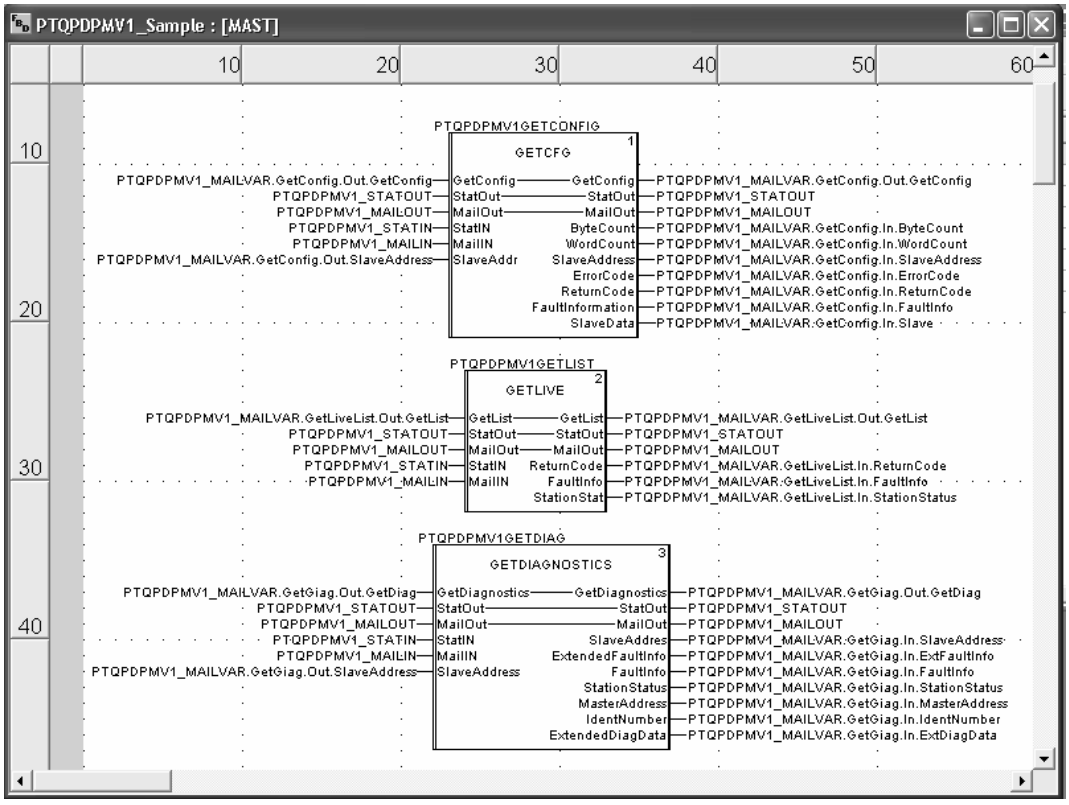


3.2 Function Blocks Operation Overview

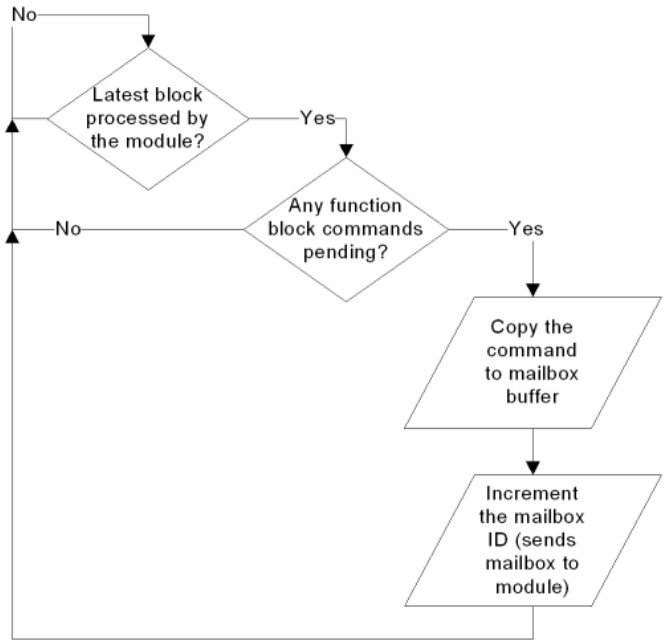
Function blocks define software components or modules that perform a specific function. Each function block has its own, pre-defined, set of inputs and outputs.

The function blocks provided with the PTQ-PDPMV1 module contains the logic to handle PROFIBUS acyclic mailbox messages and alarms. It transfers data between the main output/input mailbox arrays and the corresponding slave devices.

The PTQ-PDPMV1 module is ready to receive a mailbox message from the processor when all Function blocks have been called in the main program, which is provided in the sample.



```
MailOut[0] = StatIn.MailBoxData_LastOutMailboxMessageID
```



Each mailbox data structure is implemented through variables that are divided into "Out" and "In" data structures, where:

Out = values copied from the processor to the module

In = values copied from the module to the processor

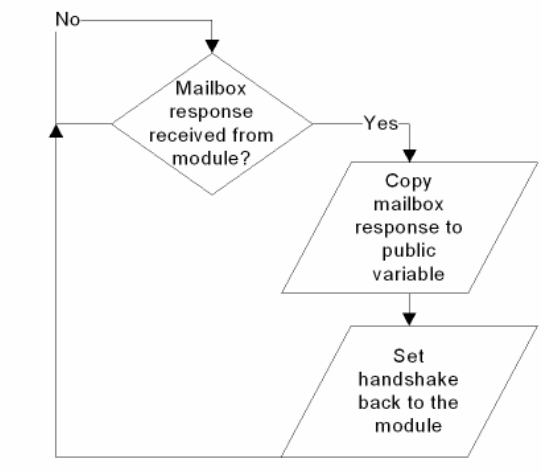
Each Out data structure contains a Cmd bit. After the Cmd bit is toggled, the logic will increment the mailbox ID (output) to send the mailbox request to the module.

The following illustration shows the interface for the SetOperatingMode mailbox:

SETOPERATINGMODE		SETOPERATINGMODE
Out		SETOPERATINGMODEOUT
SelectOperate	**0**	**BYTE**
SelectStop	**0**	**BYTE**
SelectClear	**0**	**BYTE**
ConfRequired	**0**	**BYTE**
In		SETOPERATINGMODEIN
Mode	**0**	**BYTE**
ConfRequired	**0**	**BYTE**
FaultInformation	**0**	**WORD**
StartSlaves		STARTSLAVES

The following condition indicates that the module has a mailbox response to be sent to the processor. Therefore, the function block implementation will handle the block by copying the response data to the appropriate mailbox data structure.

```
StatOut.ModuleStatus_LastinMailBoxMessageID <>
StatIn.MailBoxData_CurrentInMailboxControlIndex
```

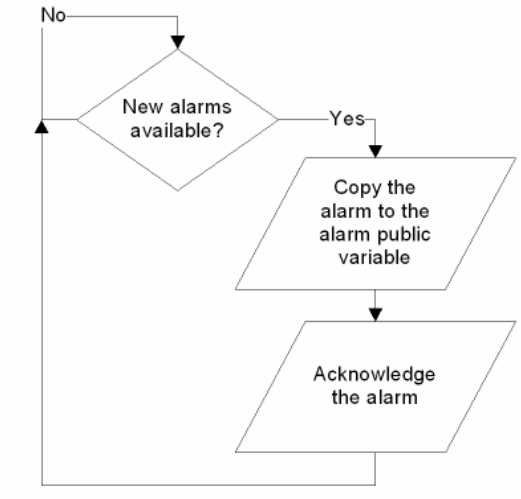


The module will increment StatIn.*MailBoxData_CurrentAlarmControlIndex* when the module contains alarms to be sent to the processor. The function block implementation will then copy the alarm to the appropriate data structure. The function block implementation uses the following expression to verify if any alarms are available:

```
StatOut.ModuleStatus_LastAlarmControlindex <>
StatIn.MailBoxData_CurrentAlarmControlIndex
```

After the alarm is copied, the logic then updates the alarm index for handshaking purposes:

```
StatOut.ModuleStatus_LastAlarmControlindex :=
StatIn.MailBoxData_CurrentAlarmControlIndex
```



Please refer to ***Mailbox Messaging*** (page 112) for further information about each mailbox parameter.

The following section provides examples of data structure groupings.

3.3 Derived Function Blocks Overview

The Unity Pro XL programming language for Schneider Electric Automation Quantum processors support user defined function blocks (DFB). The user function block types (Derived Function Blocks) are developed by the user using one or more languages (according to the number of sections). These languages are:

- Ladder language
- Structured Text language
- Instruction List language
- Functional block language FBD

A DFB type can have one or more instances where each instance is referenced by a name (symbol), and possesses DFB data types.

Derived Function blocks defined by Unity Pro XL software are entities containing:

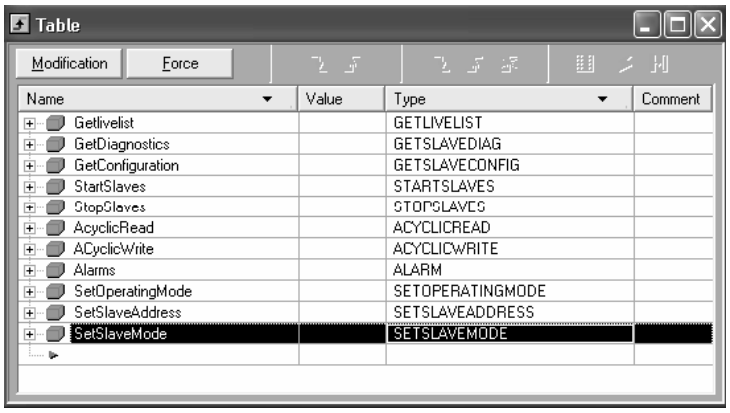
- Input and output variables acting as an interface with the application
- A processing algorithm that operates input variables and completes the output variables
- Private and public internal variables operated by the processing algorithm.

3.3.1 Using the Derived Function Blocks

To simplify programming procedures, ProSoft Technology has included a Unity Pro XL XFM Functional Module used for communication with the PTQ-PDPMV1 module. The Functional Module provides easy access to the master's cyclic and acyclic data. Specific mailbox acyclic commands are also provided to perform functions such as "Get Live List" and "Get Slave Diagnostics", and to perform "Freeze" and "Sync" commands, and others.

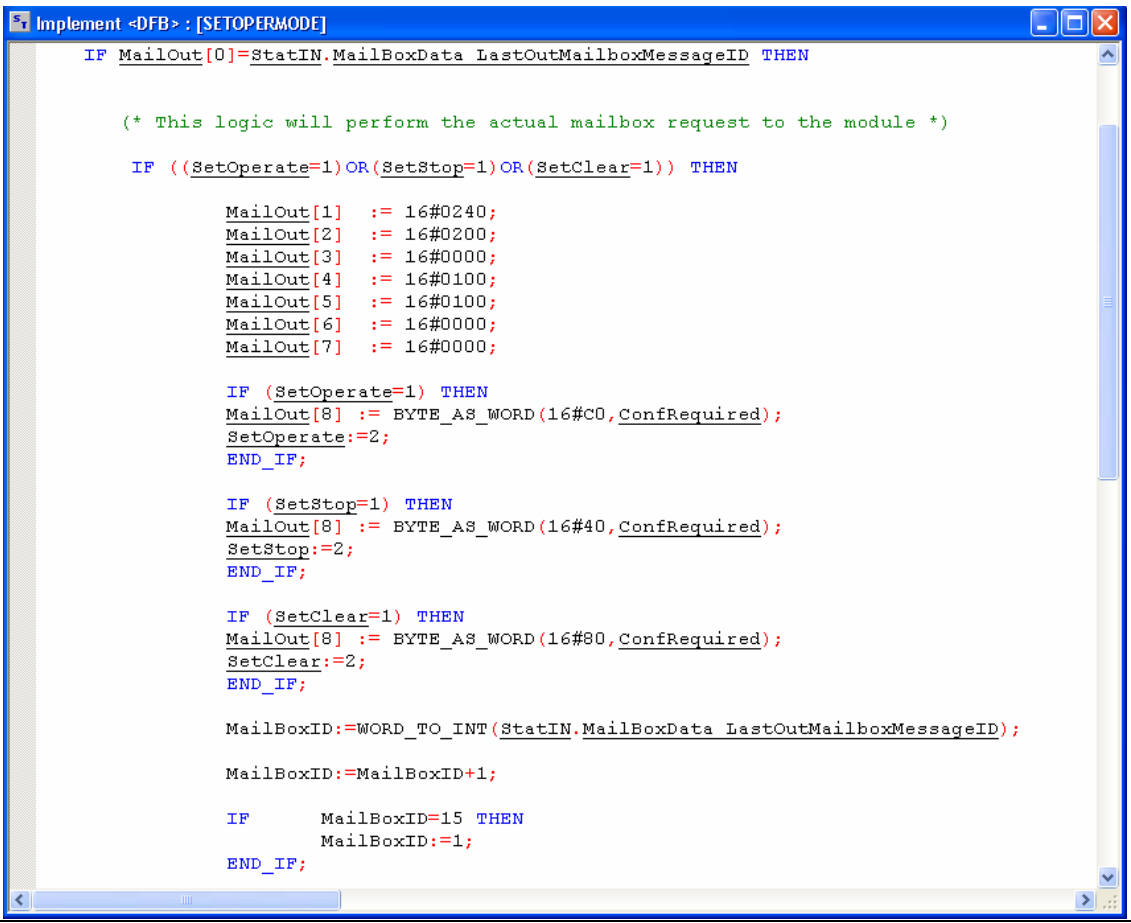
Note: It is not intended within this reference manual to include in depth programming information. You should, therefore, be familiar with IEC Function Block programming and Unity Pro XL programming language.

The PTQ_PDPMV1_Sample functional Module supports inputs and output variables used for PTQ status, acyclic mailbox and slave cyclic I/O data. All input information is located in the <Inputs> - StatIn, MailIn and DataIn area (data delivered to the Unity processor) and all output information is located in the <input/output> - StatOut, MailOut, and DataOut (data sent to the PTQ module). You can access the supported mailboxes in the provided table.



Every mailbox has its own function Block that has a unique "Impl" ST derived FB type file.

The following illustration shows part of the function block implementation (structured text code) that performs the mailbox request after the command register is triggered by the processor application. For example, the SetOperatingMode command is executed when the "Out (SetOperate,SetStop,SetClear).Cmd" bit is true.



This bit is accessed and controlled in a tag in the provided table.

Note: Refer to ***Special Function Mailbox Messaging Commands*** (page 114) for more information about Mailbox Commands.

Mailbox data values are pre-defined for the specific mailbox command.

SETOPERATINGMODE		SETOPERATINGMODE
Out		SETOPERATINGMODEOUT
SelectOperate	0	BYTE
SelectStop	0	BYTE
SelectClear	0	BYTE
ConfRequired	0	BYTE
In		SETOPERATINGMODEIN
Mode	0	BYTE
ConfRequired	0	BYTE
FaultInformation	0	WORD

The first statement `MailOut[1] := 16#0240;` represents the Message Information of the command (4002h) Set Operating Mode (see specific mailbox command).

Note: The information is byte swapped for PTQ master module (Motorola big-endian format.)

The remaining values [2] to [7] set the Command, Data Size, Frame Count, Frame Number, Offset High and Low byte header information. Again, these values are pre-defined and controlled by the FB.

Most mailbox commands have response information. Refer to ***Mailbox Messaging*** (page 112) for more information. The response information will be written to the

area of the SetOperatingMode mailbox area. This information can be read after the mailbox is received and confirm by the ID information contained in the CurrentMailboxControlIndex value.

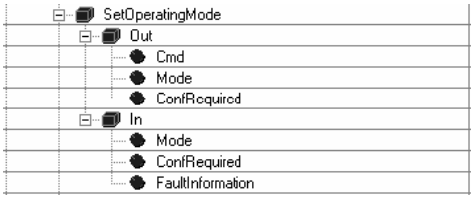
```
IF StatOut.ModuleStatus_LastinMailboxMessageID<>
   StatIN.MailBoxData_CurrentInMailboxControlIndex THEN
```

When this statement is true and the Set Operating Command was executed the following code will be executed:

```
IF MailIn[2] = 16#0200 THEN

   WORD_AS_BYTE (MailIn[8],Mode ,ConfRequired);
   FaultInformation:= MailIn[15];
```

The appropriate return value(s) for Set Operating Mode can now be read or accessed in the In.Mode, ConfRequired and FaultInformation values.



Each mailbox command can be executed and responded to using similar procedures as outlined above.

3.4 Using Mailbox Function Blocks

Function blocks define software components or modules that perform a specific function. Each function block has its own, pre-defined, set of inputs and outputs.

The function blocks provided with the PTQ-PDPMV1 module contain the logic to handle acyclic mailbox messages and alarms. It transfers data between the main output/input mailbox arrays and the corresponding slave devices.

3.4.1 Overview

The mailbox function blocks build mailbox requests to the module and read the mailbox response from the module. These mailbox function blocks are optional, meaning that the project will update PROFIBUS data and status information even if no function blocks are used.

3.4.2 Configuration

The mailbox function block contains inputs, outputs and inputs/outputs pins that must be associated to specific variables.

The mailbox function blocks (except the GetAlarm, which will be covered later) require the usage of the following pins (common for all mailbox function blocks):

Pin Name	Pin Type	Description
StatIn	Input	Input Status pin. Must be associated to the imported variable PTQPDPMV1_STATIN. It contains the status transferred from the module allowing the mailbox function block to receive the acknowledgment that the mailbox request was processed by the module. It is used also to check if a new mailbox response is available. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
MailIn	Input	Input Mailbox pin. Must be associated to the imported variable PTQPDPMV1_MAILIN. It contains the mailbox response message that is handled by the function block according to its mailbox ID. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
StatOut	Input/Output	Output Status pin. Must be associated to the imported variable PTQPDPMV1_STATOUT. It is used to check if a new mailbox response is available. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).

Pin Name	Pin Type	Description
MailOut	Input/Output	Output Mailbox pin. Must be associated to the imported variable PTQPDPMV1_MAILOUT. This variable stores the mailbox output variable that is updated from the function block when a new mailbox request is performed to the module. It consists on an array of words. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
"Trigger"	Input/Output	Move a value of 1 to this register to initiate the mailbox request. A request can only be initiated if its current value is 0 and all triggers for the other mailbox function blocks also has a value of 0. The actual name for this trigger register will be specific for each mailbox function block.

3.4.3 Trigger (BYTES)

The mailbox requests are initiated by the "trigger" (BYTE) that is defined as an input/output pin. The mailbox request is initiated after the application moves a value of 1 to the trigger Bytes.

The current value of all mailbox triggers for your application must be equal to 0 in order to perform a mailbox request. If you are using more than one mailbox function block, you must add a program code to guarantee that this condition is satisfied. Therefore, a mailbox function block is only allowed to send a new mailbox request after the processor receives the confirmation that the previous mailbox request was acknowledged by the module. This condition is determined by checking the status of all trigger Bytes (0=OK). It also prevents an attempt to send more than one mailbox request in a single PLC scan.

The following table shows the trigger Bytes used for each mailbox function block:

Function Block Name	Description	Trigger
ACYCREAD	Acyclic Read Mailbox	Acyclicread
ACYCLWRITE	Acyclic Write Mailbox	Acyclicwrite
GETALARMS	Alarm Mailbox	-
GETCFG	Get Configuration Mailbox	GetConfig
GETDIAGNOSTICS	Get Diagnostics Mailbox	GetDiagnostics
GETLIVE	Get Live List Mailbox	GetList
SETADDRESS	Set Slave Address Mailbox	SetAddress
SETOPERMODE	Set Operating Mode Mailbox	SetOperate, SetStop, SetClear
SETSLMODE	Set Slave Mode Mailbox	SetSlaveMode
STARTSTOPSLAVE	Set start and stop slaves Dynamically	StartSlaves, StopSlaves
COLDBOOT	Remote Coldboot from PLC	ColdBoot

The Get Alarm function block does not require a trigger because this mailbox is initiated from the module. Refer to ***Alarm Indication*** (page 129) for more information.

The trigger Bytes is a variable that can assume different states as follows:

Value	Description
0	OK to send new mailbox request. The last mailbox request was already acknowledged by the module.
1	Mailbox request to be performed. The Quantum program should make sure that the required conditions are satisfied (as previously discussed) before moving a value of 1 to the trigger register. The function block will then build the mailbox request by copying all mailbox input parameters to the mailbox output variable that is transferred to the module through the backplane. Then the function block will automatically change the trigger's value to 2.
2	Processor has performed the mailbox request and is waiting for the acknowledgment from the module. The acknowledgment informs that the module has received the request (the actual mailbox response is actually sent later). After the acknowledgment is received the function block will reset the trigger's value back to 0.

3.4.4 Specific Input Pins

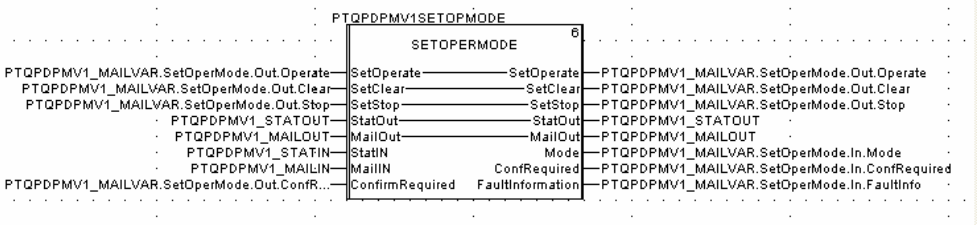
Each function block has input pins specifically for each mailbox. For example, in order to send a Get Diagnostics mailbox, the application must set the PROFIBUS slave address input pin. The processor program must configure the input pins before performing the mailbox request by moving a value of 1 to the mailbox trigger.

3.4.5 Specific Output Pins

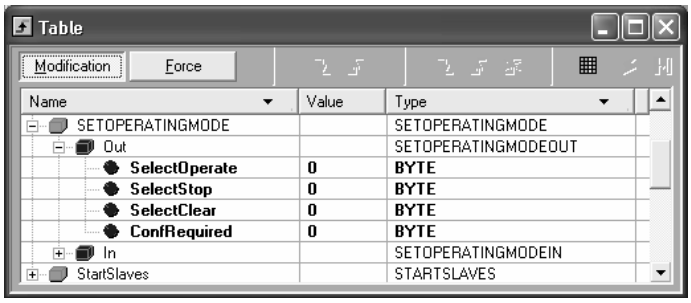
Each function block contains output pins that are updated after the mailbox response is received by the processor. For example, the Get Diagnostics function block has an *ExtendedDiagData* output pin that stores the diagnostic information received from the slave.

Example

If the Set Operating Mode mailbox function block is used as follows:



You can add SETOPERMODE to the table with three trigger variables for the Set Operating Mode mailbox. Start by moving a value of 1 to SelectStop in order to set the module's mode to STOP.

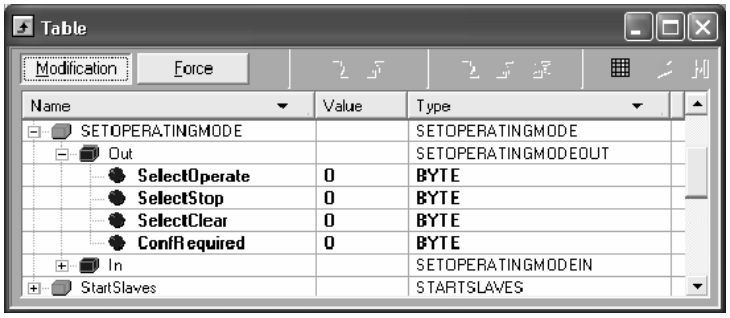


At this point you should notice the following LED display, indicating that the module's mode was changed to STOP:

LED	Status
MSTR STAT	RED
COM STAT	OFF
DBASE STAT	GREEN
TK HOLD	GREEN

You will also notice that the function block automatically clears the trigger Byte after it receives the acknowledgment from the module.

Move a value of 1 to the SelectOperate trigger Byte.



At this point you should notice the following LED display, indicating that the module's mode was changed to OPERATE:

LED	Status
MSTR STAT	GREEN
COM STAT	GREEN or OFF[1]
DBASE STAT	GREEN
TK HOLD	GREEN

The COM STAT LED will be either GREEN if the master is communicating with all slaves, blinking if it is communicating with some of the slaves or OFF if it is not communicating with any slaves.

You will also notice that the function block automatically clears the trigger register after it receives the acknowledgment from the module.

3.5 Mailbox Overview

This section provides a brief description on how to use each mailbox function block. Refer to ***Mailbox Messaging*** (page 112) for detailed information about each mailbox parameter:

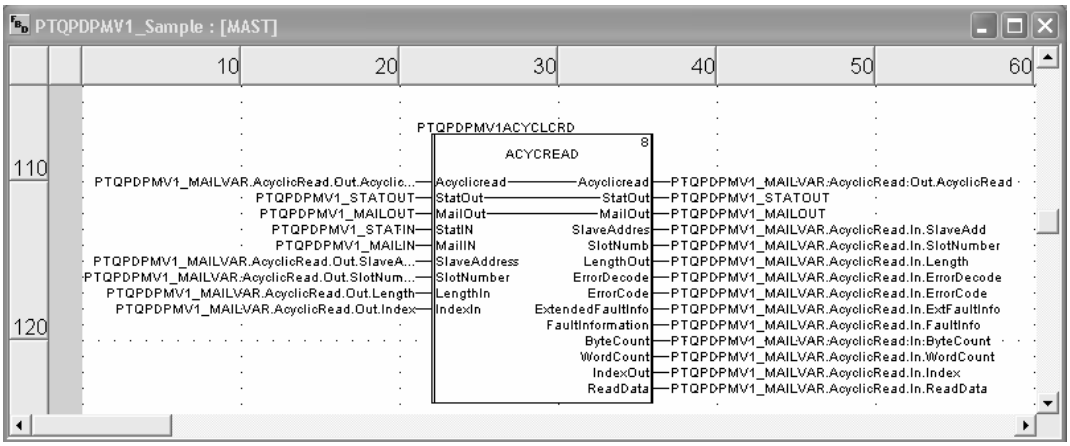
3.5.1 Acyclic Read Mailbox

Function Block: ACYCREAD

Trigger Byte: Acyclicread

Description: The ACYCREAD mailbox is used to perform an Acyclic Read request to a PROFIBUS slave device. The input pins SlaveAddress (PROFIBUS slave address), SlotNumber (slot number), IndexIn (index number) and LengthIn (length – number of bytes associated to acyclic read operation) must be configured before triggering the mailbox request. The acyclic read response data is copied to the ReadData output pin. The status information is available in the output pins (ErrorCode, ErrorDecode, ExtendedFaultInfo and FaultInformation).

The following illustration shows a sample instance of the Acyclic Read mailbox function block:



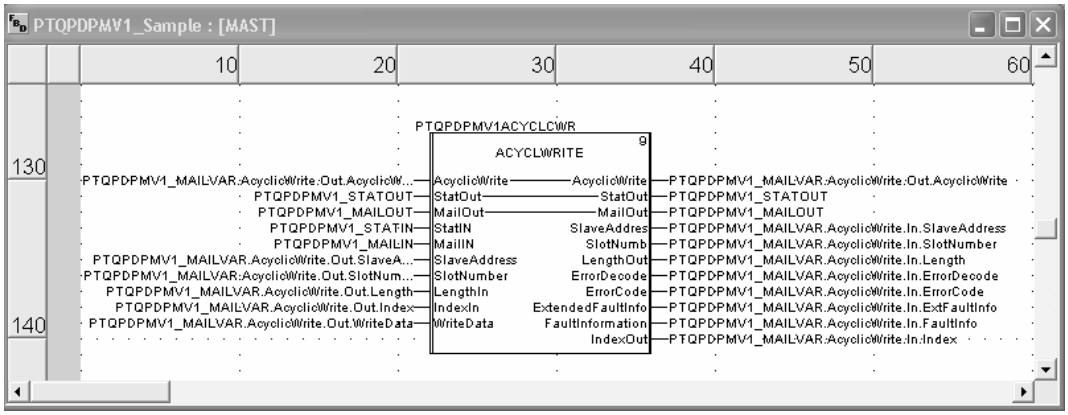
3.5.2 Acyclic Write Mailbox

Function Block: ACYCLWRITE

Trigger Byte: Acyclicwrite

Description: The ACYCLWRITE function block is used to perform an Acyclic Write request to a PROFIBUS slave device. The input pins SlaveAddress (PROFIBUS slave address), SlotNumber (slot number), IndexIn (index number) and LengthIn (length - number of bytes associated to acyclic read operation) must be configured before triggering the mailbox request. The actual data to be written to the PROFIBUS slave should be associated to the WriteData input pin. The status information is available at the output pins (ErrorCode, ErrorDecode, ExtendedFaultInfo and FaultInformation).

The following illustration shows a sample instance of the Acyclic Write mail box function.



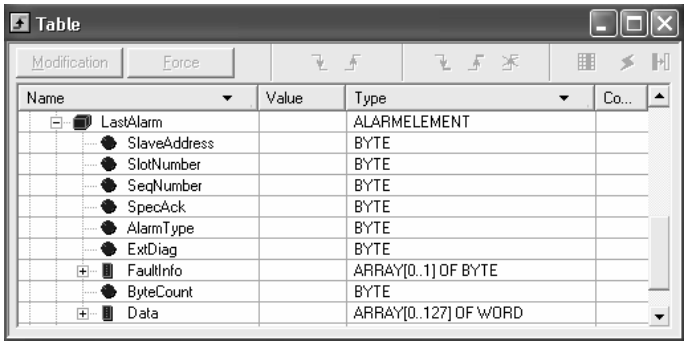
3.5.3 Alarm Mailbox

Function Block: GETALARMS

Trigger Byte: -

Description: The GETALARMS mailbox is used to read the alarm mailbox messages sent by the module. The module will automatically generate the alarm mailboxes after it receives the alarm message from the PROFIBUS slave. Therefore no triggers are required for this mailbox.

The last alarm received is copied at the *LastAlarm* output pin. This is a data structure that contains all alarm information:



This function block also keeps track of the last 100 alarms through the HistAlarm output pin.

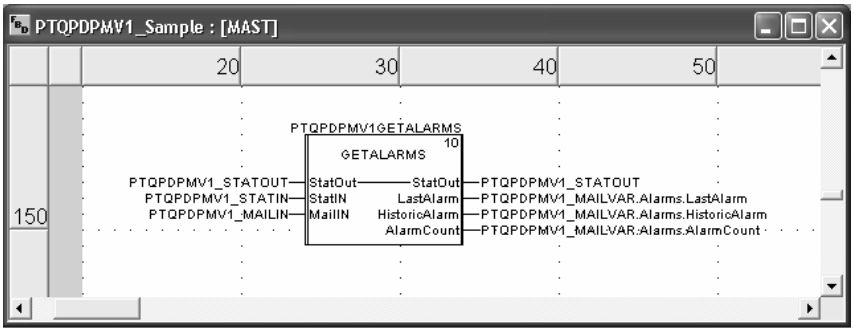
For example, if the module receives 100 alarms (first alarm - Sequence Number = 1, second alarm - Sequence Number = 2, and so on), after alarm #100 is received, the processor application could refer to these alarms stored at the following output pins:

```
Last Alarm - Alarm #100
HistAlarm[1] - Alarm # 99
HistAlarm[2] - Alarm # 98
HistAlarm[3] - Alarm # 97
HistAlarm[4] - Alarm # 96
HistAlarm[5] - Alarm # 95
HistAlarm[6] - Alarm # 94
HistAlarm[7] - Alarm # 93
HistAlarm[8] - Alarm # 92
HistAlarm[9] - Alarm # 91
HistAlarm[10] - Alarm #90
```

If the HistoricAlarm buffer is full and it receives a new alarm, then the oldest alarm in the queue will be deleted to reserve space for the new alarm.

The AlarmCount output pin is incremented every time the alarm mailbox is received. This register will rollover at 30000. The processor application can keep track of this register to determine when the processor has received a new alarm mailbox message from the module.

The following illustration shows a sample instance of the GetAlarms mailbox function block:

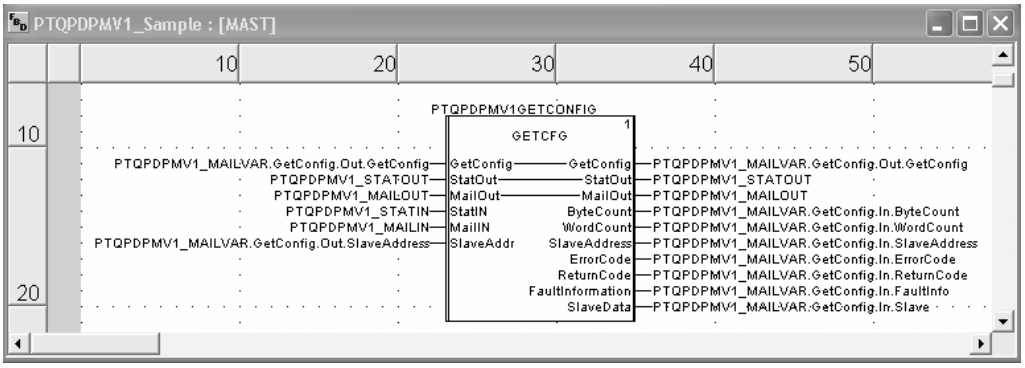


3.5.4 GetConfiguration Mailbox

Function Block: GETCFG

Trigger Byte: GetConfig

Description: The GETCFG function block can be used to read the configuration of any PROFIBUS slave connected to the PTQ-PDPMV1 module. The SlaveAddr input pin must be configured with the PROFIBUS slave address of the PROFIBUS device. The configuration data is stored at the SlaveData output pin. The byte count of the slave configuration is stored at ByteCount output pin. The ErrorCode, ReturnCode and FaultInformation output pins can be used for status verification.



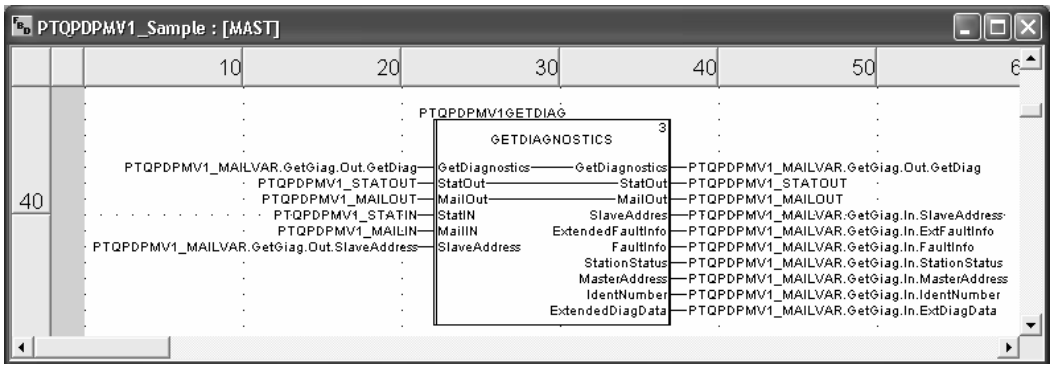
3.5.5 GetDiagnostics Mailbox

Function Block: GETDIAGNOSTICS

Trigger Byte: GetDiagnostics

Description: The GETDIAGNOSTICS function block can be used to read the diagnostics from any PROFIBUS slave connected to the PTQ-PDPMV1 module. The slave address must be set at the SlaveAddress input pin. The diagnostics data is copied at the ExtendedDiagData output pin. The number of bytes of the

diagnostics message is stored at the ByteCount output pin (status+identification+extended diagnostics). The ExtendedFaultInfo and FaultInformation output pins can be used for status information. The master address is stored at the MasterAddress output pin.



3.5.6 GetLiveList Mailbox

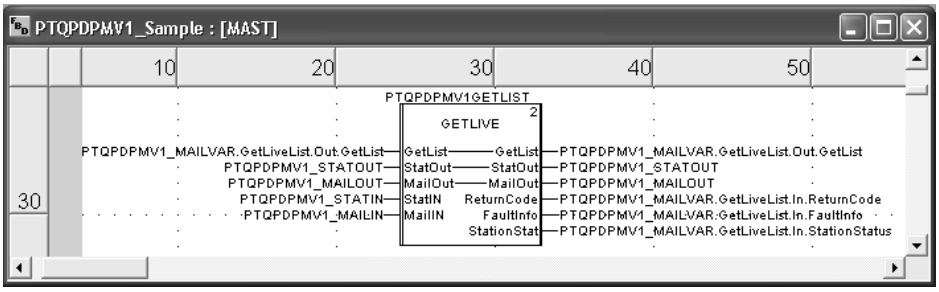
Function Block: GETLIVE

Trigger Byte: GetList

Description: The GETLIVE function block can be used to read the live list from the module containing the status of each device at the PROFIBUS network. The live list is stored at the StationStatus output pin. The live list data is an array of bytes stored as follows:

```
StationStatus [0] - status of device configured with PROFIBUS address 0
StationStatus [1] - status of device configured with PROFIBUS address 1
StationStatus [2] - status of device configured with PROFIBUS address 2
StationStatus [3] - status of device configured with PROFIBUS address 3
StationStatus [4] - status of device configured with PROFIBUS address 4
Etc…
```

Refer to ***Mailbox Messaging Error Codes*** (page 133) for further information about the valid status codes. The ReturnCode and FaultInformation output pins can be used for mailbox status information.

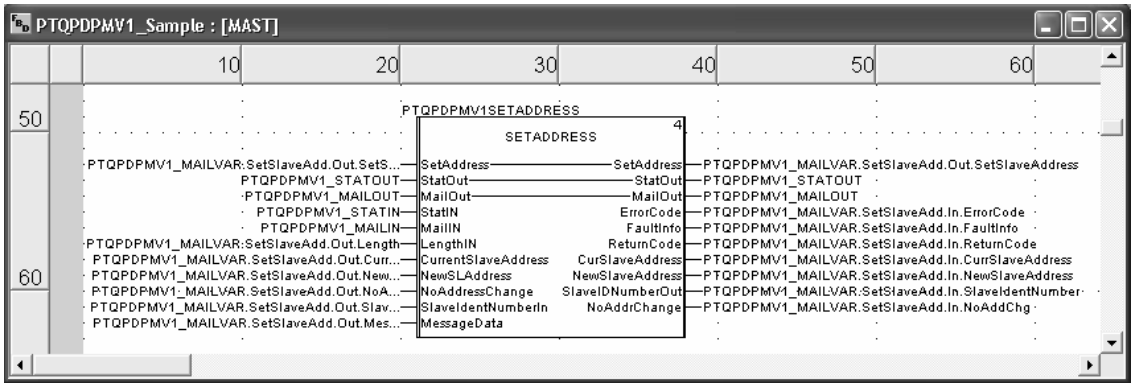


3.5.7 SetSlaveAddress Mailbox

Function Block: SETADDRESS

Trigger Byte: SetAddress

Description: The SETADDRESS mailbox can be used to change the slave address. Only specific PROFIBUS devices support this feature. The application must set the CurrentSlaveAddress (current address) and NewSLAddress (new address) input pins. It is also possible to deliver user data through the MessageData input pin (the number of bytes must be set through the LengthIn input pin). The SlaveIdentNumberIn input pin must be set with the Ident number for the slave. The FaultInformation output pin can be check for mailbox status information.



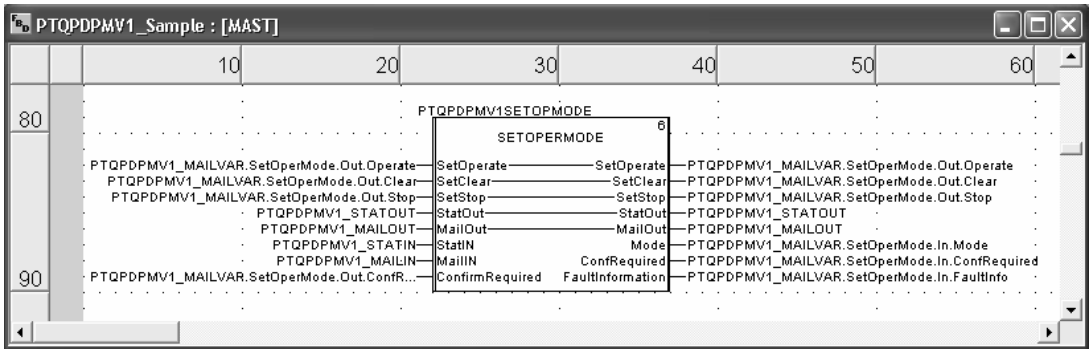
3.5.8 SetOperatingMode Mailbox

Function Block: SETOPERMODE

Trigger Byte: SetOperate, SetStop, SetClear

Description: The following trigger values can be used to change the current operating mode of the module:

```
SETOPERATINGMODE.Out.SelectOperate  = Set Operate
SETOPERATINGMODE.Out.SelectStop     = Set Stop
SETOPERATINGMODE.Out.SelectClear    = Set Clear
```

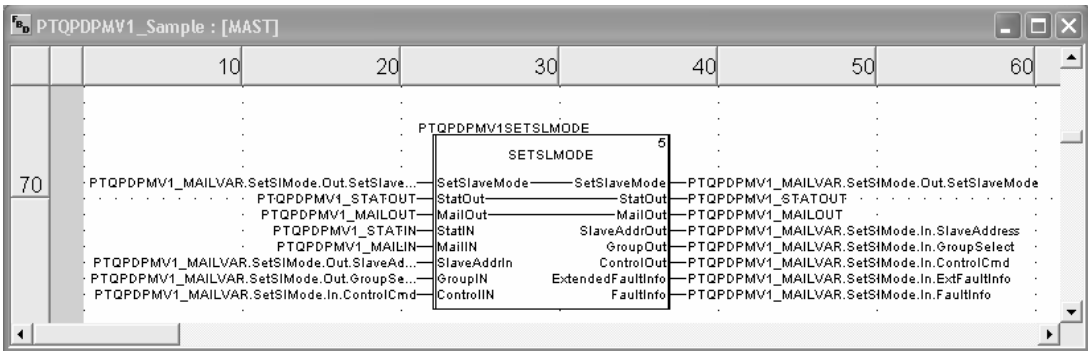


3.5.9 SetSlaveMode Mailbox

Function Block: SETSLMODE

Trigger Byte: SetSlaveMode

Description: The SETSLMODE function block can be used to request the module to sync, unsync, freeze or unfreeze. The slave address must be selected through the SlaveAddrIn input pin. If the operation is directed to a group of slaves then the Group number must be set through the GroupIn input pin parameter. The actual code that will select the operation type must be configured through the ControlIn input pin. Please check the User Manual for the valid control codes.



Important Note: The next mailbox is only to be used with Anybus firmware version 3.50 and later. Earlier released versions do not support this feature.

To determine your Anybus firmware version, use ProSoft Configuration Builder to connect to the module and open the Diagnostics window. On the Main Menu, press [3] to view the Control Registers. Note the firmware version number displayed on this screen.

```
Control Registers:
Boot Loader Ver.  = 2.05
Serial Number     = FFFFFFFF
Firmware version  = 3.80
Module Status     = 0401    (0x0401=Running with slaves)
```

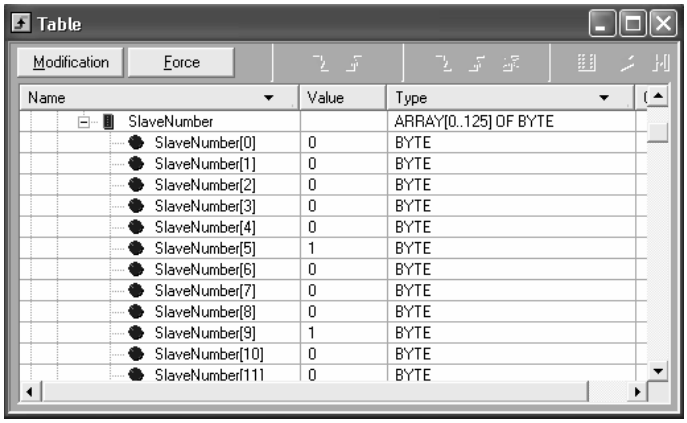
3.5.10 StartStopSlaves Mailbox

Function Block: STARTSTOPSLAVES

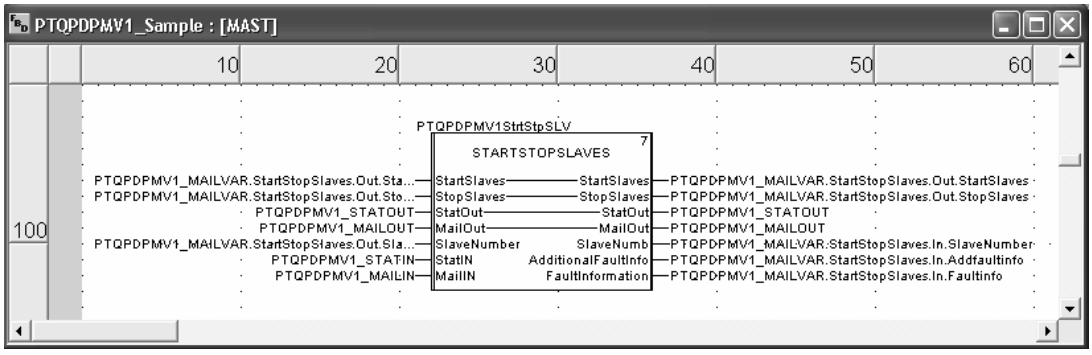
Trigger Byte: StartSlaves, StopSlaves

Description: The STARTSTOPSLAVES function block can be used to request the module to start or stop certain slaves dynamically. The slave address must be selected through the SlaveNumber input pin. This is an array of 126 slaves. Change the value for a specific slave from 0 to 1 to start or stop communication with the master.

The following illustration shows that when you execute the mailbox command, Slave #5 and Slave #9 will start communicating with the master.



You can confirm the execution of the mail box by verifying that the SlaveNumb output pin exactly matches the SlaveNumber Input pin.



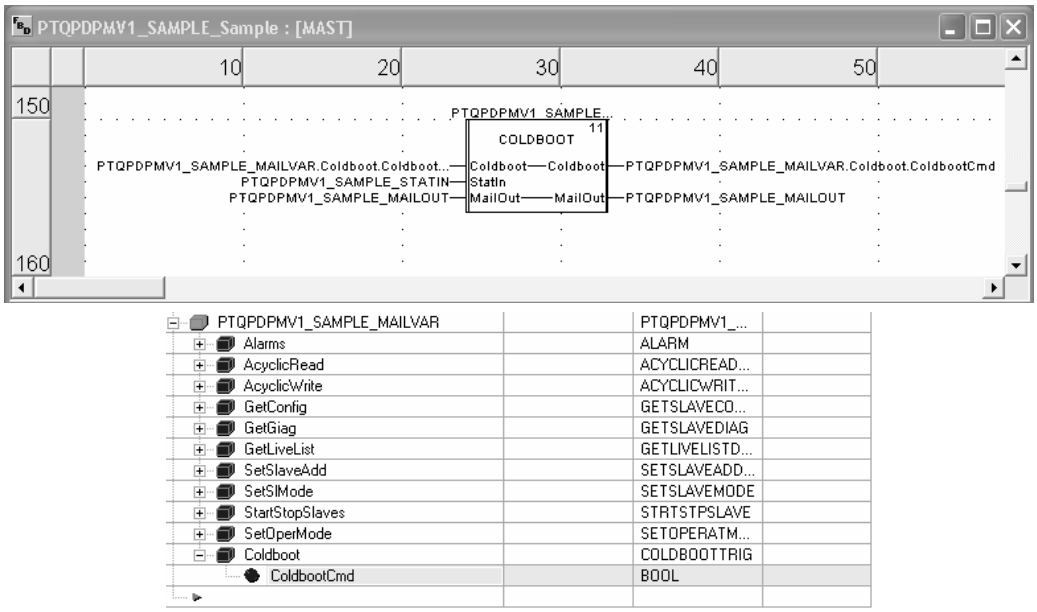
3.5.11 Coldboot Mailbox

Important: The Coldboot mailbox is only supported on PTQ-PDPMV1 modules running firmware version 1.19 or newer. Earlier versions of the firmware do not support this feature. If you require this functionality, please contact ProSoft Technical Services for information on how to upgrade your module.

Function Block: COLDBOOT

Trigger Byte: ColdBoot

Description: The COLDBOOT function block allows you to remotely reboot the module. To trigger a reboot the module, change the value of the Coldboot bit from 0 (zero) to 1 (one). The bit is reset back to 0 when the function is executed.



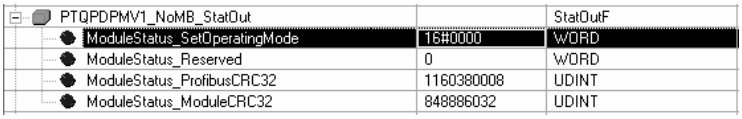
HSBY Note: This function block will reset both the active master and the passive master.

Using a Control Word to Reboot the Module

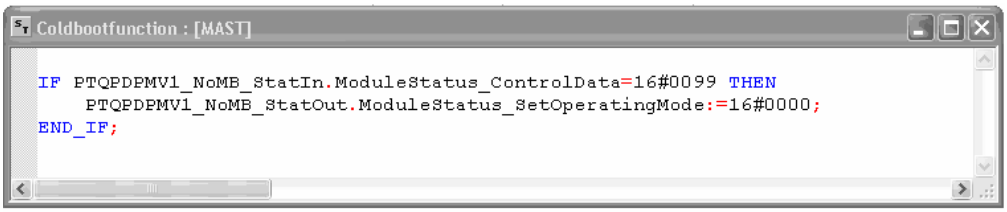
If you need to coldboot the module without using the Coldboot mailbox, use the ModuleStatus_SetOperatingMode control word.

➢ ***To reboot the module:***

1 Enter the value 16#9999 (Hex) in the ModuleStatus_SetOperatingMode register, as shown in the following illustration.



2 Add the following lines to the program file:



This logic will reset the value in ModuleStatus_SetOperatingMode to 16#0000 (Hex).

Note: It is normal for the Passive Master in Hot Standby applications to reboot twice using this procedure.

4 Configuring the Processor with Concept

In This Chapter

- Overview 73
- Before You Begin 74
- Information for Concept Version 2.6 Users 75
- Step 1 - Exporting the Files from PCB 76
- Step 2 - Convert the Function Blocks 78
- Step 3 - Setup the Concept Project 81
- Step 4 - Import the Variables 83
- Step 5 - Create the Function Block Instances 86
- Step 6 - Download the Concept Project 92
- Using the Concept Project 92
- Using Mailbox Function Blocks 95
- Mailbox Overview 100

HSBY Note: Concept software does not support 140CPU67160 processor and therefore does not support the PTQ-PDPMV1 HSBY functions.

Important Note: Concept software does not report whether the PTQ module is present in the rack, and therefore is not able to report the health status of the module when the module is online with the Quantum processor. Please take this into account when monitoring the status of the PTQ module.

4.1 Overview

This section will guide you through the steps required to setup your Concept Project with the PTQ-PDPMV1 module. There are a total of 6 steps required as follows:

Step 1 - *Export the Files from PCB* **(page 76)**

This step shows how to export the required files from PCB (containing function blocks, variables and data type definitions) that will be used during this procedure.

Step 2 - *Convert the Function Blocks* **(page 78)**

The .ASC function blocks (exported at step 1) must be converted before used in the Concept project. This step shows how to convert the function blocks from .ASC to .DFB format.

Step 3 - *Setup the Concept Project* **(page 81)**

This step shows how to setup the Concept Project and configure the required amount of processor memory for your application.

Step 4 - *Import the Variables* **(page 83)**

This step shows how to import the variables into your Concept project by using the .txt file (obtained at step 1). The PCB configuration will determine the addressing of the variables.

Step 5 - *Create the Function Block Instances* **(page 86)**

This step shows how to create an instance of the function blocks that were converted at step 2. It also shows that some function block pins must be linked with the variables that were imported at step 4.

Step 6 - *Download the Project to the Quantum processor* **(page 92)**

Once you download your project to the Quantum processor the procedure is completed.

After you followed these steps you can refer to the following topics for more information on how to perform basic tasks:

Using the Concept Project

This section shows how to access PROFIBUS data and status information through the imported variables. It also shows how to perform a mailbox request from the processor.

Mailbox Overview

This section provides a general overview of the mailbox function blocks that are supplied as a sample application.

4.2 Before You Begin

1. Verify that your PC has the following software tools installed:
   - ProSoft Configuration Builder (version 2.0.0 Build 15 or later)
   - Concept Programming Unit
2. Create a folder *C:\project\DFB*, where:
   C:\project – will store the main Concept project (.PRJ)
   C:\project\DFB – will store the data type definition file (.DTY) and the function blocks that will be used by the Concept project.

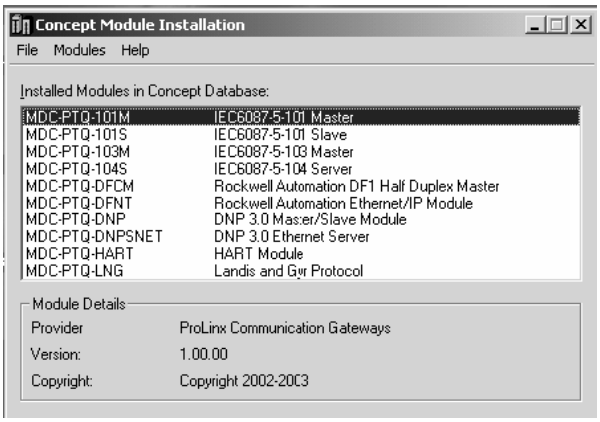
Warning: The Function Block is intended for new installations of PTQ-PDPMV1. If you have an existing installation, the following procedure will overwrite your settings, and may cause loss of functionality. DO NOT overwrite a working application until you have thoroughly reviewed the following topics.

4.3 Information for Concept Version 2.6 Users

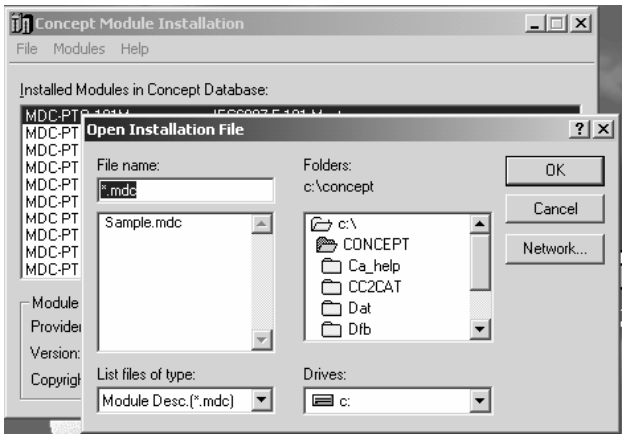
This guide uses Concept PLC Programming Software version 2.6 to configure the Quantum PLC. The ProTalk installation CD includes MDC module configuration files that help document the PTQ installation. Although not required, these files should be installed before proceeding to the next section.

4.3.1 Installing MDC Configuration Files

1 From a PC with Concept 2.6 installed, choose **Start** → **Programs** → **Concept** → **ModConnect Tool**.
This action opens the Concept Module Installation dialog box.

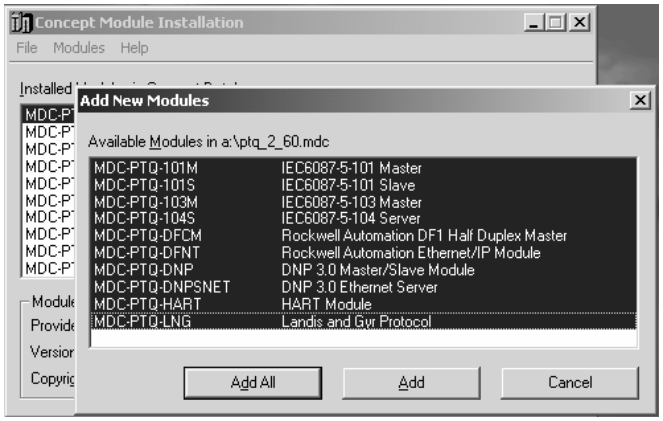


2 Choose **File** → **Open Installation File**.
This action opens the Open Installation File dialog box:



3 If you are using a Quantum processor, you will need the MDC files. In the File/Open dialog box, navigate to the **MDC Files** directory on the ProTalk CD.
4 Choose the MDC file and help file for your version of Concept:
- Concept 2.6 users: select PTQ_2_60.mdc and PTQMDC.hlp
- Concept 2.5 users: select PTQ_2_50.mdc and PTQMDC.hlp.

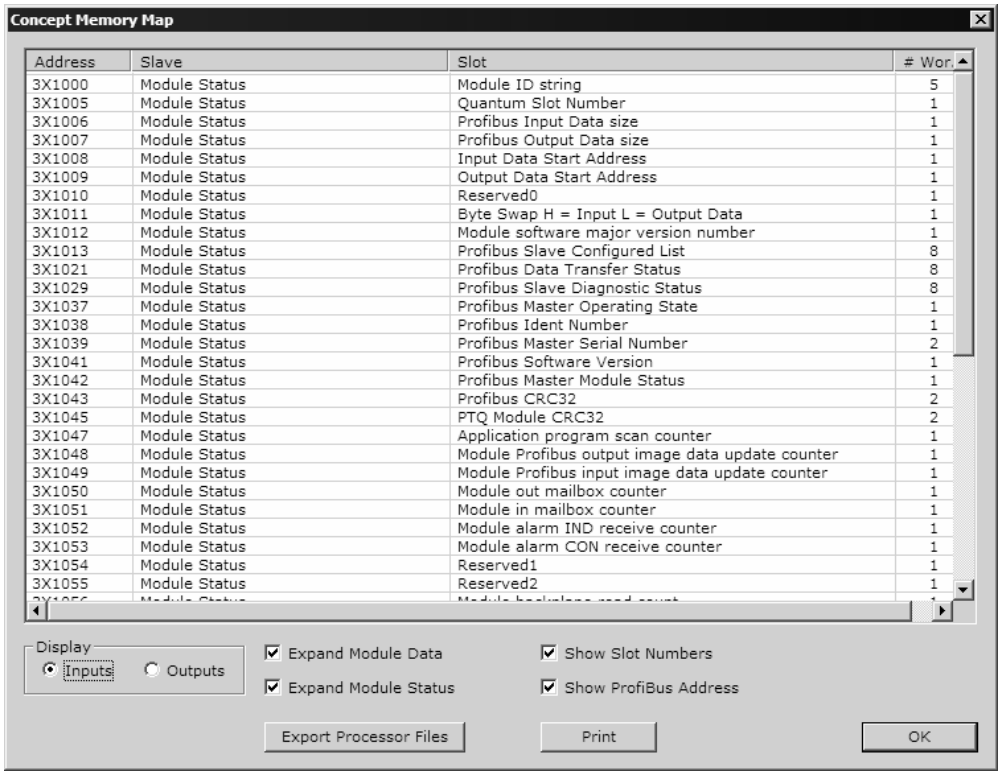
Select the files that go with the Concept version you are using, and then click **OK**. This action opens the Add New Modules dialog box.



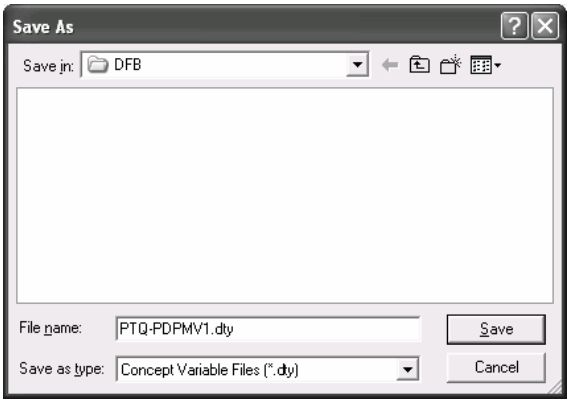
5 Click the **Add All** button. A series of message boxes may appear during this process. Click **Yes** or **OK** for each message that appears.
6 When the process is complete, open the File menu and choose Exit to save your changes.

4.4 Step 1 - Exporting the Files from PCB

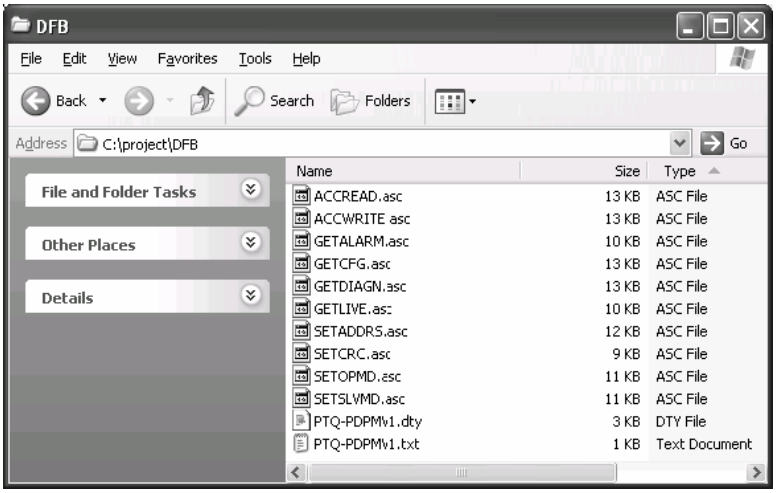
1 In ProSoft Configuration Builder, right-click the PROFIBUS DP folder, and then click *Configure*.
2 Click *Show Concept Map*
3 Click *Export Processor Files*


Concept Memory Map
Address Slave Slot # Wor
3X1000 Module Status Module ID string 5
3X1005 Module Status Quantum Slot Number 1
3X1006 Module Status Profibus Input Data size 1
3X1007 Module Status Profibus Output Data size 1
3X1008 Module Status Input Data Start Address 1
3X1009 Module Status Output Data Start Address 1
3X1010 Module Status Reserved0 1
3X1011 Module Status Byte Swap H = Input L = Output Data 1
3X1012 Module Status Module software major version number 1
3X1013 Module Status Profibus Slave Configured List 8
3X1021 Module Status Profibus Data Transfer Status 8
3X1029 Module Status Profibus Slave Diagnostic Status 8
3X1037 Module Status Profibus Master Operating State 1
3X1038 Module Status Profibus Ident Number 1
3X1039 Module Status Profibus Master Serial Number 2
3X1041 Module Status Profibus Software Version 1
3X1042 Module Status Profibus Master Module Status 1
3X1043 Module Status Profibus CRC32 2
3X1045 Module Status PTQ Module CRC32 2
3X1047 Module Status Application program scan counter 1
3X1048 Module Status Module Profibus output image data update counter 1
3X1049 Module Status Module Profibus input image data update counter 1
3X1050 Module Status Module out mailbox counter 1
3X1051 Module Status Module in mailbox counter 1
3X1052 Module Status Module alarm IND receive counter 1
3X1053 Module Status Module alarm CON receive counter 1
3X1054 Module Status Reserved1 1
3X1055 Module Status Reserved2 1
Display
Inputs
Outputs
Expand Module Data
Expand Module Status
Show Slot Numbers
Show ProfiBus Address
Export Processor Files
Print
OK

4 Browse to the folder *C:\project\DFB* and click OK.



5 All the files required for your Concept application will be now located at the *C:\project\DFB*.


DFB
File Edit View Favorites Tools Help
Back Search Folders
Address C:\project\DFB
Go
File and Folder Tasks
Other Places
Details
Name Size Type
ACCREAD.asc 13 KB ASC File
ACCWRITE asc 13 KB ASC File
GETALARM.asc 10 KB ASC File
GETCFG.asc 13 KB ASC File
GETDIAGN.asc 13 KB ASC File
GETLIVE.asc 10 KB ASC File
SETADDRS.asc 12 KB ASC File
SETCRC.asc 9 KB ASC File
SETOPMD.asc 11 KB ASC File
SETSLVMD.asc 11 KB ASC File
PTQ-PDPMv1.dty 3 KB DTY File
PTQ-PDPMv1.txt 1 KB Text Document

The following section provides a general overview of the files that were exported:

4.4.1 -.ASC files

Each function block is available in ASCII format. These files can be converted through the Concept Converter tool in order to be used in the Concept project.

File Name	Description	Required/Optional[1]
ACCREAD.ASC	Acyclic Read Mailbox	Optional
ACCWRITE.ASC	Acyclic Write Mailbox	Optional
GETALARM.ASC	Alarm Mailbox	Optional
GETCFG.ASC	Get Configuration Mailbox	Optional
GETDIAGN.ASC	Get Diagnostics Mailbox	Optional
GETLIVE.ASC	Get Live List Mailbox	Optional
SETADDRS.ASC	Set Slave Address Mailbox	Optional
SETCRC.ASC	Set CRC (not mailbox – used to sync CRCs)	Required
SETOPMD.ASC	Set Operating Mode Mailbox	Optional
SETSLVMD.ASC	Set Slave Mode Mailbox	Optional

File Name	Description	Required/Optional[1]
SETSLVS.ASC	Start/Stop Slaves Dynamically	Optional
ColdBT.ASC	Remote Coldboot from PLC	Optional

[1]Optional means that you should import this function block only if your application requires that specific mailbox. The status and PROFIBUS data will be available even if no optional mailbox function blocks are imported into your project. The SETCRC.ASC function block is required to synchronize the input and output ***CRCs*** (page 94) while configuring the module.

4.4.2 -.DTY file

This file contains the data type definitions that will be required for the Concept project. The file name depends on the project name selected at PCB (default is PTQ-PDPMV1.dty).

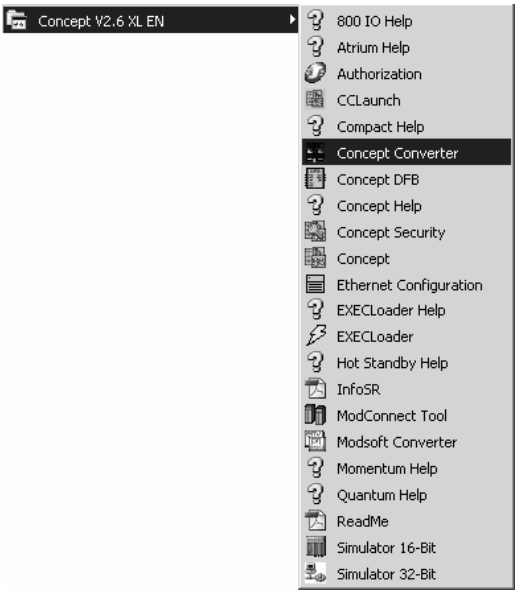
Note: If your application requires multiple PTQ-PDPMV1 modules for the same project, you must merge the .DTY files. Refer to ***Using Multiple PTQ-PDPMV1 Modules with Concept*** (page 228) for instructions.

4.4.3 -.TXT file

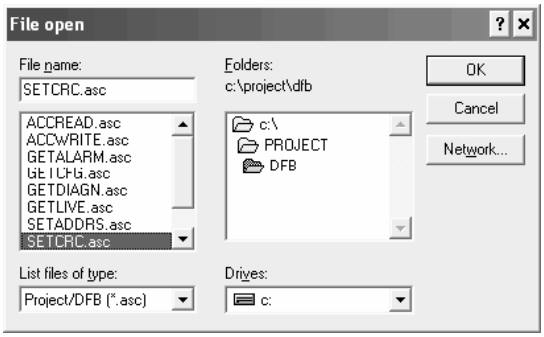
This file contains the variables that will be imported to Concept. The default file name is the file name depends on the project name selected at PCB (default is PTQ-PDPMV1.txt)

4.5 Step 2 - Convert the Function Blocks

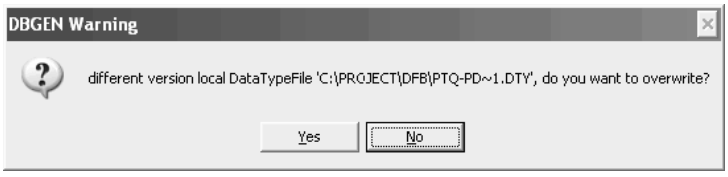
1 Run the *Concept Converter* tool as follows:



2 Click *File-Import*. Browse the *SETCRC.ASC* file at the *C:\project\DFB* folder



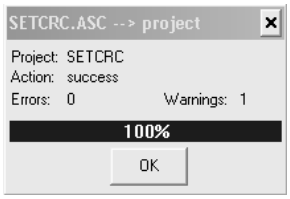
After you click OK the following warning message will be displayed. Click **No**.



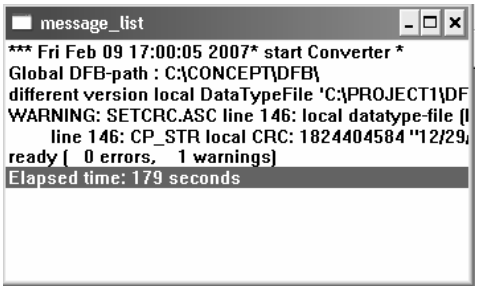
3 If you use the **same filename** as the ones PCB generated (PTQ-PDPMV1_sample) you will see the following message:

DBGEN Warning
local datatype-file (PTQ-PD~1.dty) does not match datatypes in DFB/project
restart Converter and click 'yes' to overwrite datatype-file
OK

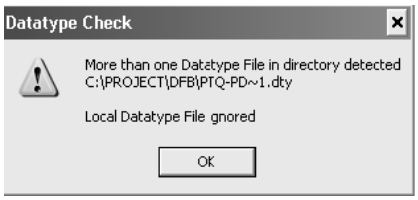
Click OK to dismiss the warning message.



Next, you will see the following message box:


message_list
*** Fri Feb 09 17:00:05 2007* start Converter *
Global DFB-path : C:\CONCEPT\DFB\
different version local DataTypeFile 'C:\PROJECT1\DF
WARNING: SETCRC.ASC line 146: local datatype-file (l
line 146: CP_STR local CRC: 1824404584 "12/29/
ready (0 errors, 1 warnings)
Elapsed time: 179 seconds

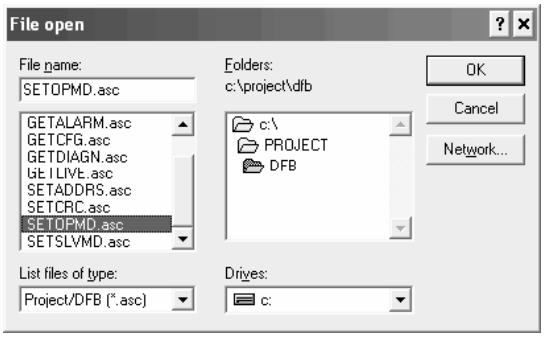
This message does not indicate an error condition, because the ASC files were built with a different DTY file, which was overwritten with the new DTY file exported by PCB. The new DTY file has all I/O information for cyclic data and (optional) slave diagnostics.
If you use a **different filename** than the one generated by PCB (for example Processline1_Master) you will see the following message when you attempt to import the new ASC file.



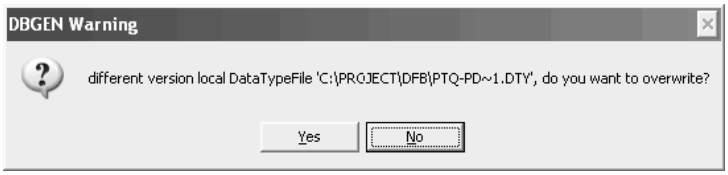
Click OK to dismiss the message. After importing all .ASC files, delete the old DTY file (PTQ-PD~1.dty). The old DTY file does not contain any Cyclic I/O data.

4 List all optional mailbox functions that will be required for your application. Repeat steps 2 and 3 for each required mailbox. This setup procedure will consider (as an example) that only the Set Operating Mode mailbox is required. So the following steps (5 and 6) will repeat the same procedure for the Set Operating Mode mailbox function block:

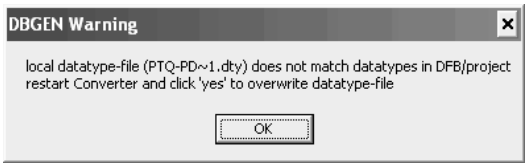
5 Click *File-Import*. Browse the *SETOPMD.ASC* file at the *C:\project\DFB* folder



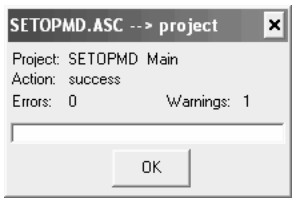
After you click OK the following warning message will be displayed. Click **No**.



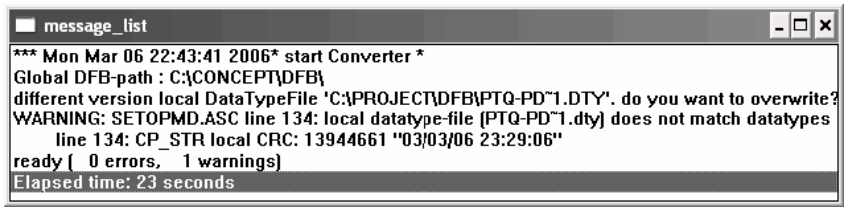
Then the following warning message is displayed. Click OK.


DBGEN Warning
local datatype-file (PTQ-PD~1.dty) does not match datatypes in DFB/project
restart Converter and click 'yes' to overwrite datatype-file
OK

6 After the import procedure is concluded the following window is displayed. Click *OK*:



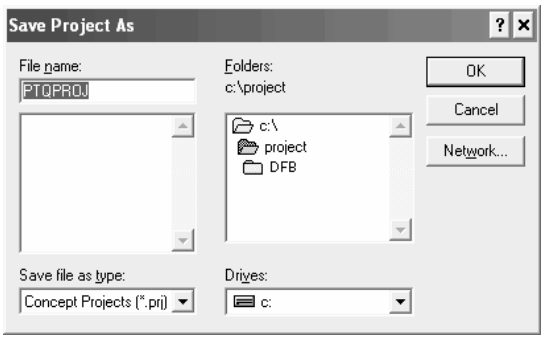
The following warning is displayed (this is expected).


message_list
*** Mon Mar 06 22:43:41 2006* start Converter *
Global DFB-path : C:\CONCEPT\DFB\
different version local DataTypeFile 'C:\PROJECT\DFB\PTQ-PD~1.DTY'. do you want to overwrite?
WARNING: SETOPMD.ASC line 134: local datatype-file (PTQ-PD~1.dty) does not match datatypes
line 134: CP_STR local CRC: 13944661 "03/03/06 23:29:06"
ready (0 errors, 1 warnings)
Elapsed time: 23 seconds

7 At Concept Converter click *File-Exit*.

4.6 Step 3 - Setup the Concept Project

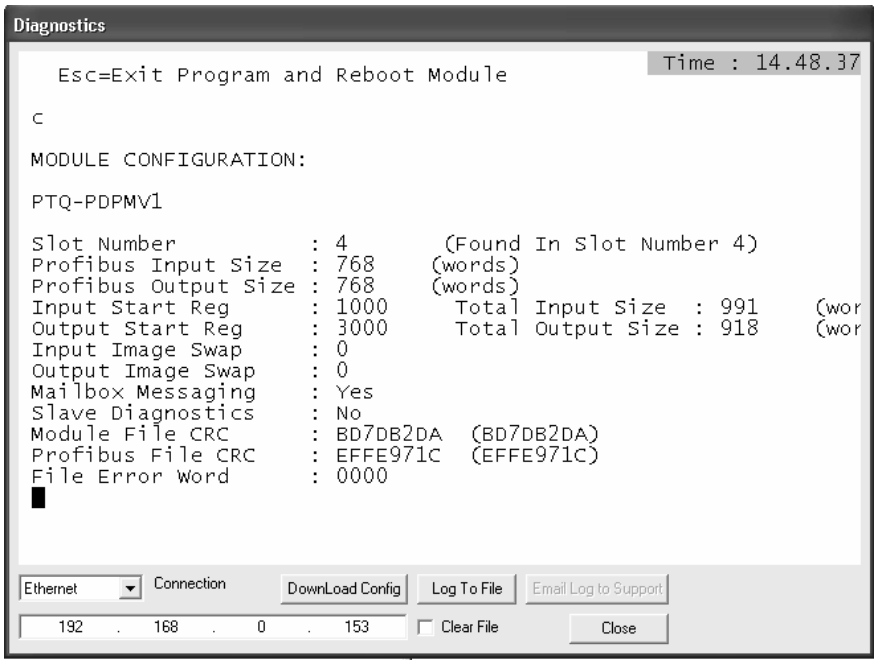
1 Start Concept Version 2.6.
2 Open the File menu, and then choose New Project.
3 Open the File menu again, and then click Save Project As…
4 Navigate to *C:\project,* and enter PTQPROJ as the file name. Click OK to save the file.



5 Configure the general settings for your application. Select the correct Quantum processor type (PLC Selection) and other modules that will be located at the Quantum rack.
6 In PLC Configuration double-click PLC Memory Partition. Verify that the number of input registers and output registers are sufficient for your application.

Note: You can view the number of input and output words required for your PTQ-PDPMV1 application in the Diagnostics window in ProSoft Configuration

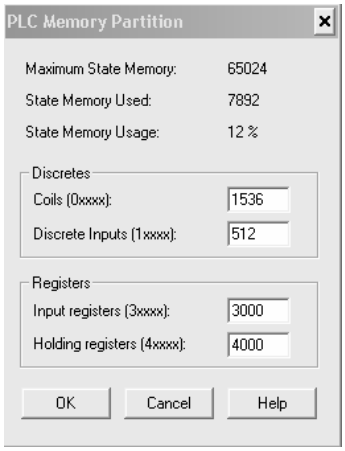
Builder. From the module's configuration/debug menu, press [B] to open the Block Transfer Statistics menu.


Diagnostics
Time : 14.48.37
Esc=Exit Program and Reboot Module
C
MODULE CONFIGURATION:
PTQ-PDPMV1
Slot Number : 4 (Found In Slot Number 4)
Profibus Input Size : 768 (words)
Profibus Output Size : 768 (words)
Input Start Reg : 1000 Total Input Size : 991 (wor
Output Start Reg : 3000 Total Output Size : 918 (wor
Input Image Swap : 0
Output Image Swap : 0
Mailbox Messaging : Yes
Slave Diagnostics : No
Module File CRC : BD7DB2DA (BD7DB2DA)
Profibus File CRC : EFFE971C (EFFE971C)
File Error Word : 0000
Ethernet
Connection
DownLoad Config
Log To File
Email Log to Support
192 . 168 . 0 . 153
Clear File
Close

Using the example in the illustration, note the following values:

	Total Size	Start Address	Last Address
PROFIBUS Input	991	301000	301990
PROFIBUS Output	918	403000	403917

For this example, select 3000 input registers and 4000 holding registers as shown in the following illustration.



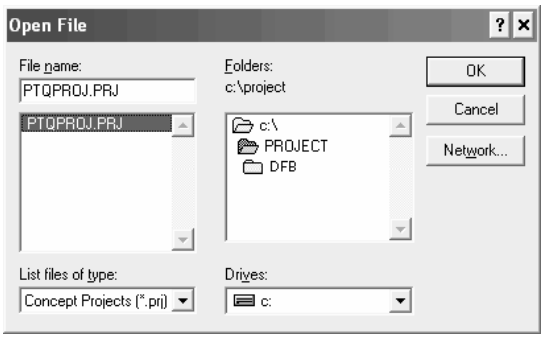
Note: Use these values for reference only. The illustration above indicates that you can only use 2000 registers, because the start register is at 1000 and the

count is 3000. If your input requires more than 2000 registers, refer to the following paragraph.

Important: You must configure the number of registers required for your application correctly, otherwise the backplane driver will not transfer any data between the processor and the module. Please note that the Input Data Size and Output Data Size parameters configured in PCB will configure only the number of registers required for PROFIBUS data. However the module will require more registers for status and mailbox transfer. For this reason, you must verify the total number of registers through the Diagnostics window.

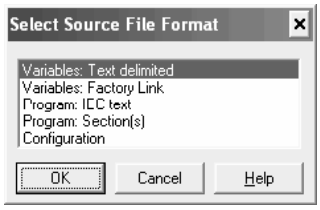
7 In Concept, open the File menu, and then choose Close Project.

8 Open the FIle menu again, and then choose Open-Project to reopen the file that you have just saved. This step allows Concept to recognize the data type definitions and function blocks that are located in *C:\project\DFB*

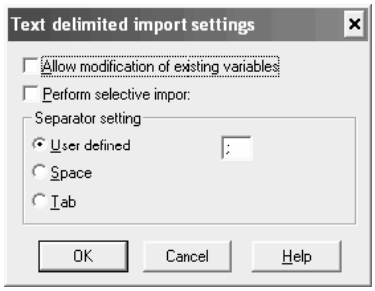


4.7 Step 4 - Import the Variables

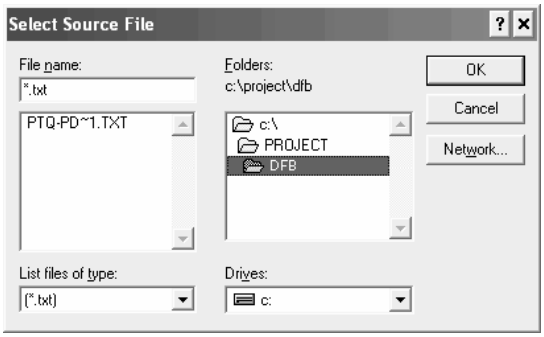
1 In Concept, open the File menu, and then choose Import. Select Variables: Text Delimited and click OK.



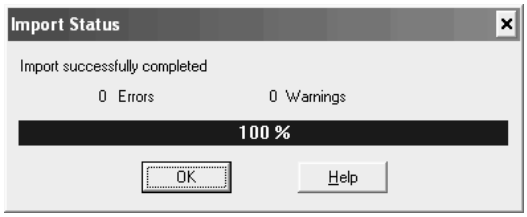
2 Select User Defined, with ";" as the delimiter, and leave all other options unselected. Click OK.



3 Navigate to the .TXT file you exported in Step 1, located in *C:\project\DFB* and then click OK to import the variables.

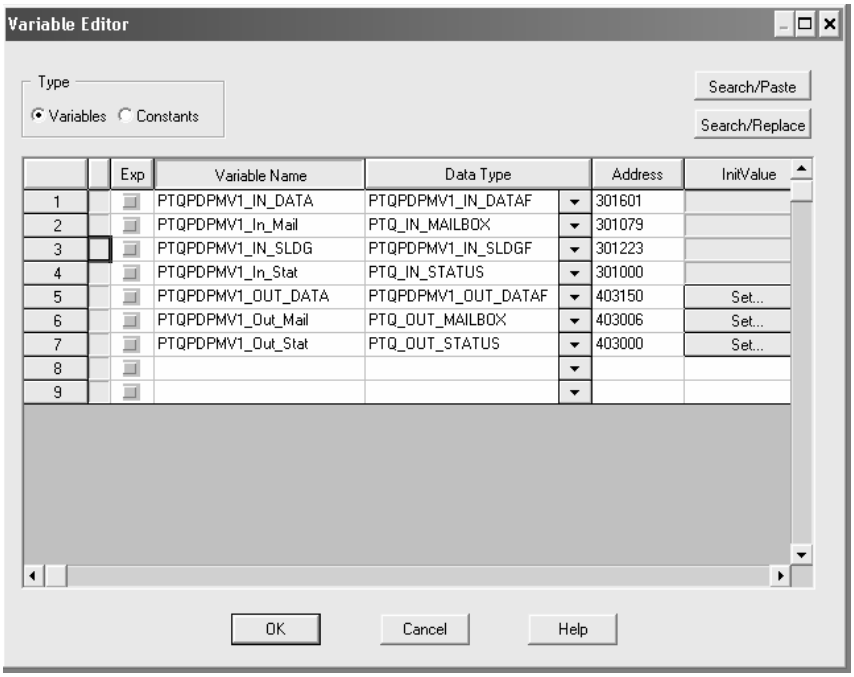


4 When the import procedure is completed, click OK to dismiss the Import Status message box.



To view the variables that were created during this procedure, open the Project menu, and then choose Variable Declarations. The following illustration shows a variable for Slave Diagnostics (PTQPDPMV1_SLDG), which is an optional selection. Refer to PTQ Input and Output Data Blocks for detailed information on the structure of these blocks and how they are affected by various configuration options.

Note: The memory addresses will match the settings configured through ProSoft Configuration Builder.



The following variables are available for your application:

Variable	Transferred From	Transferred To	Description
PTQPDPMV1_In_Stat	PTQ-PDPMV1	Quantum	Status Data
PTQPDPMV1_In_Mail	PTQ-PDPMV1	Quantum	Mailbox Buffer
PTQPDPMV1_IN_DATA	PTQ-PDPMV1	Quantum	Input PROFIBUS Data
PTQPDPMV1_Out_Stat	Quantum	PTQ-PDPMV1	Status Data
PTQPDPMV1_Out_Mail	Quantum	PTQ-PDPMV1	Mailbox Buffer
PTQPDPMV1_OUT_DATA	Quantum	PTQ-PDPMV1	Output PROFIBUS Data
PDQPDPMV1_IN_SLDG	PTQ-PDPMV1	Quantum	Slave Diagnostic Data

Status - the status data can be used to monitor the status of the module and the PROFIBUS network (input). The function blocks also use the status data for handshaking purposes during the mailbox handling (input and output).

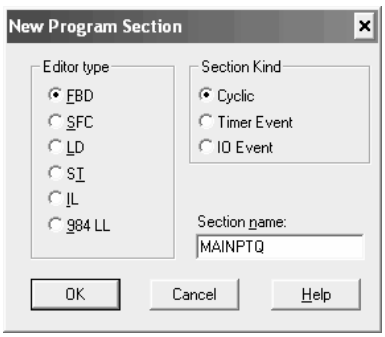
Mailbox Buffer - these variables store the mailbox requests and responses between the processor and the module. These mailbox variables (input and output) must be linked to the MailIn and MailOut function block pins (as covered later in this document).

PROFIBUS Data - these variables store the input and output PROFIBUS data associated to the PROFIBUS slaves.

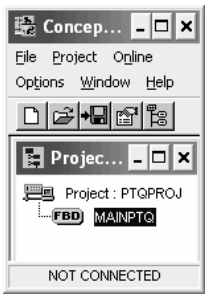
Slave Diagnostic Data - Diagnostic for every configured slave on the Profibus network will be passed from PTQ-PDPMV1 to Quantum processor

4.8 Step 5 - Create the Function Block Instances

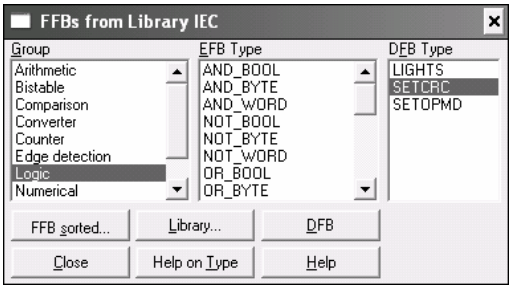
1 At Concept click Project-Project Browser
2 At Project Browser right-click at *Project: PTQPROJ* and click *New Section*
3 Configure the Program Section as follows (select the Editor Type as FBD).



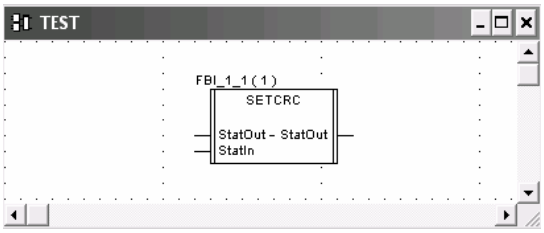
4 Double-click at the FBD section you have just created:



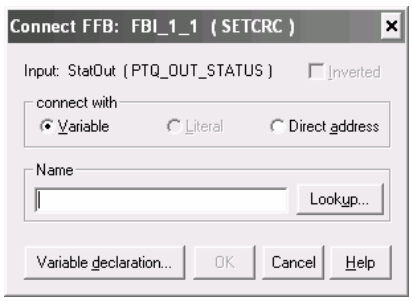
5 Click *Objects-FFB Selection*. Click at the *DFB* button and select the *SETCRC* Function Block. Click at the *Close* button to confirm.



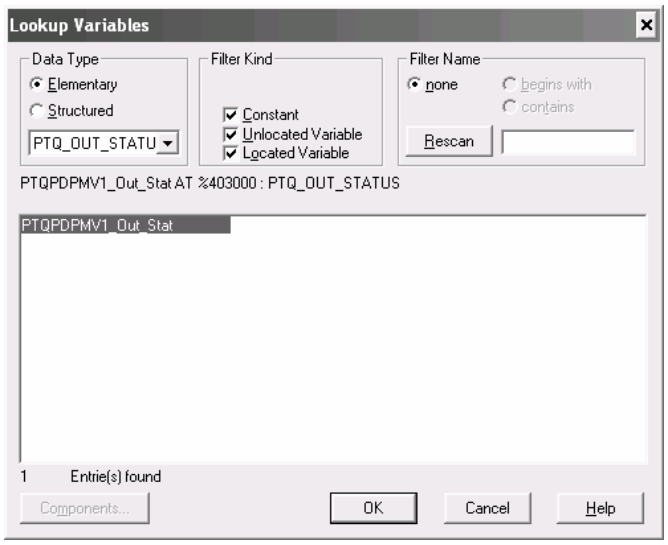
Insert the SETCRC function block at the MAINPTQ section.



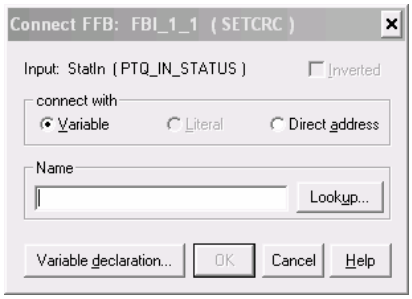
Select the imported variables to be associated with the input and output pins. The input status and output status variables must be the same ones that were previously imported by PCB.
Double-click at the StatOut pin and click LookUp…



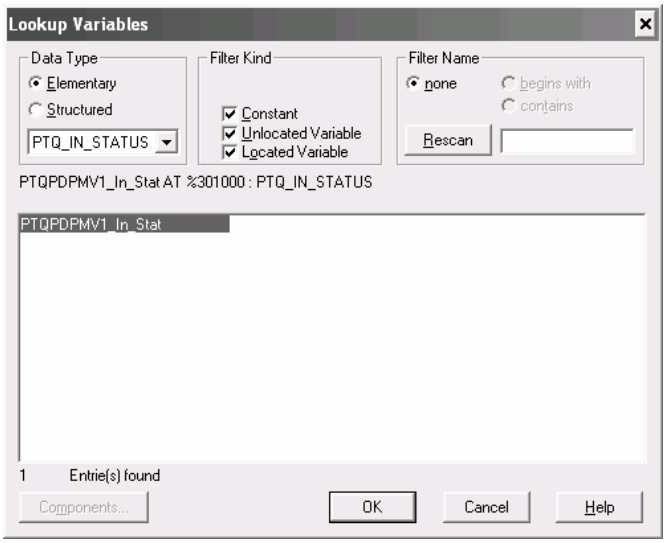
Select the existing variable that was previously imported to the project:



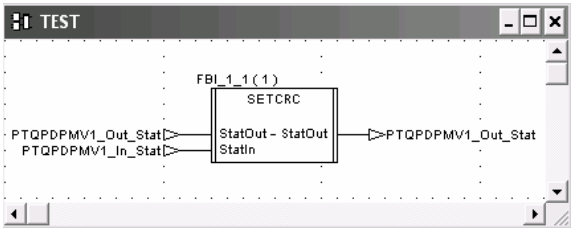
Double-click at the StatIn pin and select LookUp…



Select the variable that was previously imported to the project:

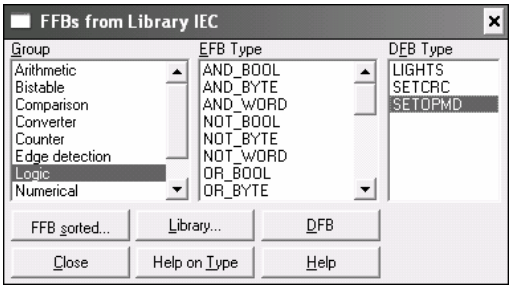


The function block is now ready:

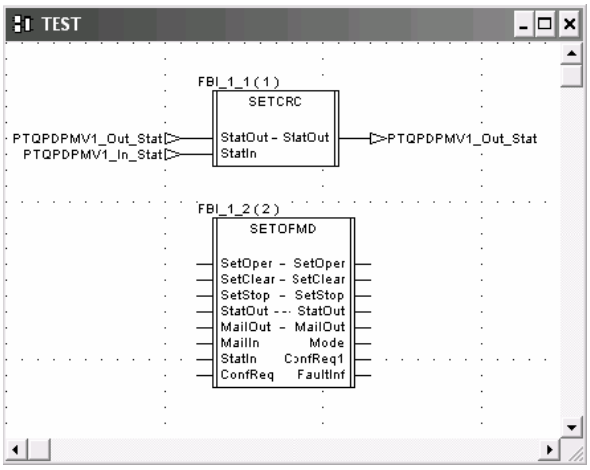


6 Now you should repeat steps 5 through 7 for every mailbox function block to be used by the application. This setup procedure considers only the SETOPMD (Set Operating Mode function block) for the next 3 steps.

7 Select *Objects-FFB Selection*. Click at the *DFB* button and select the *SETOPMD* Function Block. Click at the *Close* button to confirm.



8 Insert the SETOPMD function block at the MAINPTQ section.



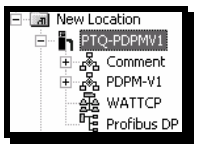
Associate the variables required for the input and output pins for this function block.

IMPORTANT: The following pins must be associated to the variables previously imported. The user cannot associate these pins to any other variables. This important step must be followed for all function blocks.

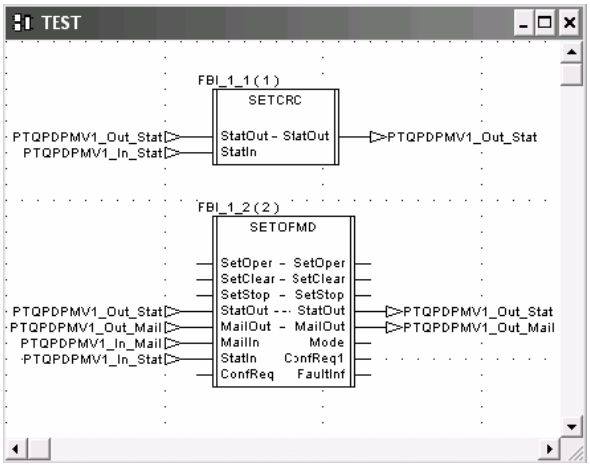
PIN	Type	Default Variable Name[1]	Data Type
StatOut	Input/output	PTQPDPMV1_Out_Stat	PTQ_OUT_STATUS
MailOut	Input/output	PTQPDPMV1_Out_Mail	PTQ_OUT_MAILBOX
StatIn	Input	PTQPDPMV1_In_Stat	PTQ_IN_STATUS
MailIn	Input	PTQPDPMV1_In_Mail	PTQ_IN_MAILBOX

The actual variable name will depend on the module name configured in PCB. The default module name is PTQPDPMV1. However the data types used by these variables will always have a fixed name as shown in the table above.

Default file name (PTQ-PDPMV1):



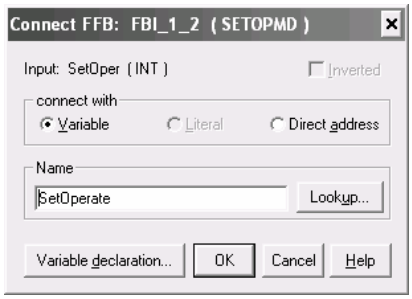
9 It is suggested to initially associate these variables to the correct pins before creating any variables for the other pins (as follows):



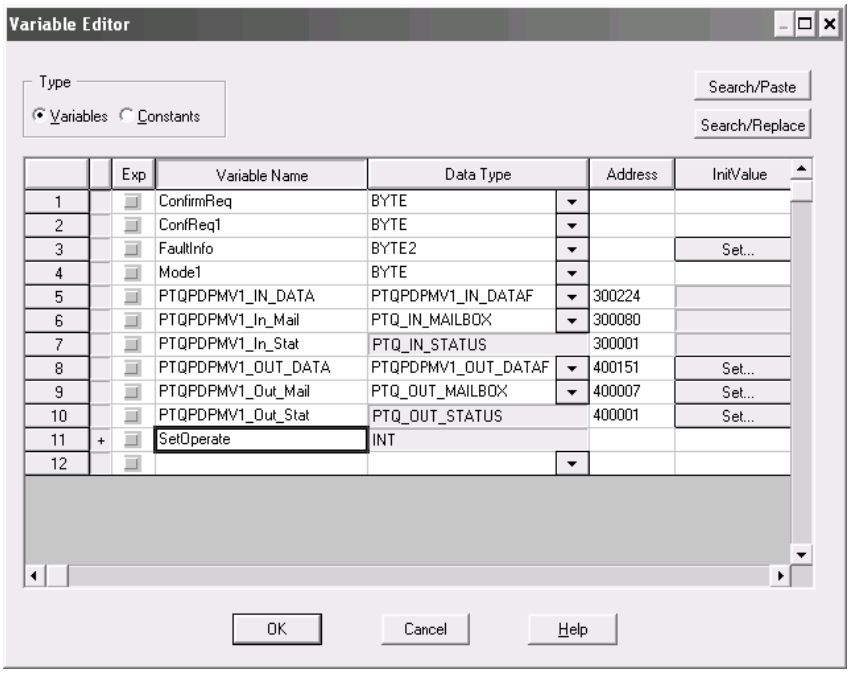
Now you can create other variables and associate these to the rest of the pins. For example, to associate a variable to the SetOper input/output pin follow the steps below:

a) Double-click at the SetOper pin

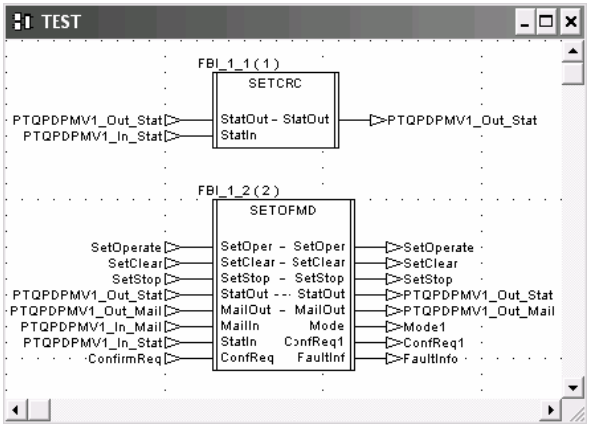
b) Choose a variable name (this example uses SetOperate) and click OK



c) Click OK to confirm the variable creation

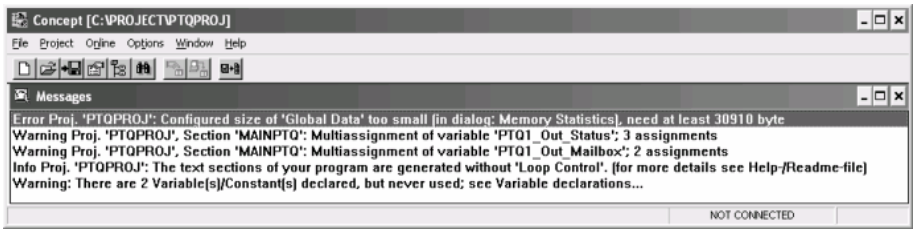


10 Repeat the same procedure for all other pins until the function block configuration is completed:



11 Save the Concept Project (File-Save Project)

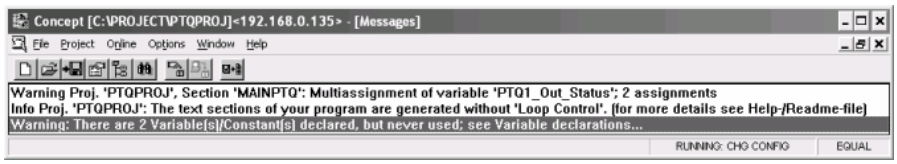
Note: While analyzing the project (depending on the number of mailbox function blocks used the following error message might be generated:


Concept [C:\PROJECT\PTQPROJ]
File Project Online Options Window Help
Messages
Error Proj. 'PTQPROJ': Configured size of 'Global Data' too small (in dialog: Memory Statistics), need at least 30910 byte
Warning Proj. 'PTQPROJ', Section 'MAINPTQ': Multiassignment of variable 'PTQ1_Out_Status'; 3 assignments
Warning Proj. 'PTQPROJ', Section 'MAINPTQ': Multiassignment of variable 'PTQ1_Out_Mailbox'; 2 assignments
Info Proj. 'PTQPROJ': The text sections of your program are generated without 'Loop Control'. (for more details see Help-/Readme-file)
Warning: There are 2 Variable(s)/Constant(s) declared, but never used; see Variable declarations...
NOT CONNECTED

The error message means that you must increase the size of the global output data for your project. You can select Project-PLC Configuration-PLC Selection to increase the size of the global output data.

4.9 Step 6 - Download the Concept Project

Download the project to the Quantum processor (Online Connect and Online-Download...). Once the download operation is concluded there will be a few warning messages generated in Concept. This is expected since it is just informing to the user that some input/output variables are being used by more than one function block.


Concept [C:\PROJECT\PTQPROJ]<192.168.0.135> - [Messages]
File Project Online Options Window Help
Warning Proj. 'PTQPROJ', Section 'MAINPTQ': Multiassignment of variable 'PTQ1_Out_Status'; 2 assignments
Info Proj. 'PTQPROJ': The text sections of your program are generated without 'Loop Control'. (for more details see Help-/Readme-file)
Warning: There are 2 Variable[s]/Constant[s] declared, but never used; see Variable declarations...
RUNNING: CHG CONFIG
EQUAL

4.10 Using the Concept Project

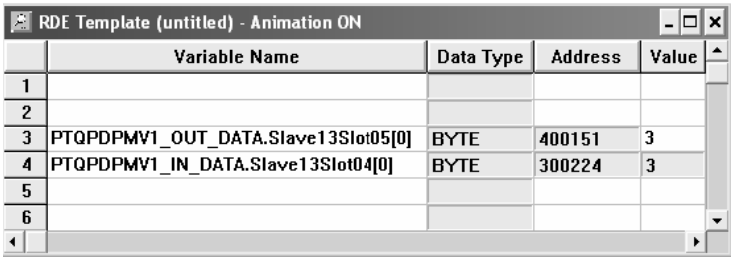
4.10.1 Accessing PROFIBUS Data

After the module's CRC values are synchronized (through the SETCRC function block) then no other function blocks are required for PROFIBUS Input/Output data exchange. You can refer to the variables that were imported to the Concept project and use either the *PTQPDPMV1_OUT_DATAF* or *PTQPDPMV1_IN_DATAF* data types. These variables contain a structure of sub variables that will store the data associated to all slaves configured at ProSoft Configuration Builder.

The following illustration shows an example where two variables are used to store the PROFIBUS input and output data.

PTQPDPMV1_OUT_DATA.Slave13Slot05[0] - Stores output byte 0 of slot 5 from the slave (PROFIBUS address 13)

PTQPDPMV1_IN_DATA.Slave13Slot04[0] - Stores input byte 0 of slot 4 from the slave (PROFIBUS address 13)


RDE Template (untitled) - Animation ON

	Variable Name	Data Type	Address	Value
1				
2				
3	PTQPDPMV1_OUT_DATA.Slave13Slot05[0]	BYTE	400151	3
4	PTQPDPMV1_IN_DATA.Slave13Slot04[0]	BYTE	300224	3
5				
6				

Note$_1$: to check if the CRC values are synchronized you can look at the status of the CFG ERR LED (Off - CRCs are synchronized, On - CRCs are not synchronized)

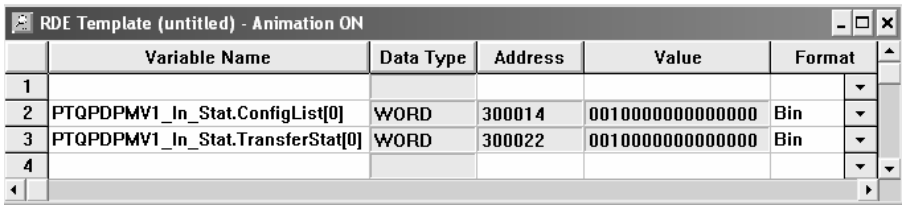
Note$_2$:The PTQPDPMV1_IN_DATA variable is the default variable name. The actual name for your application might vary depending on the module name that you have selected at PCB.

4.10.2 Accessing Status Data

The module constantly updates the status data to the processor. The status data provides general information about the module, PROFIBUS slaves and backplane status. It is automatically (no function blocks are required). Refer to ***status registers*** (page 201) for more information.

Refer to the imported variables that use the PTQPDPMV1_IN_STATUS data type for the status data.

The following illustration shows an example of two registers that inform if the first 16 devices are currently configured (PTQPDPMV1_In_Stat.ConfigList[0]), and if these devices are in data exchange mode (PTQPDPMV1_In_Stat.TransferStat[0]). The following illustration shows an example where slave 13 is configured and in data exchange mode.


RDE Template (untitled) - Animation ON

	Variable Name	Data Type	Address	Value	Format
1					
2	PTQPDPMV1_In_Stat.ConfigList[0]	WORD	300014	0010000000000000	Bin
3	PTQPDPMV1_In_Stat.TransferStat[0]	WORD	300022	0010000000000000	Bin
4					

Note[1]: The actual variable name will depend on the project name you have selected at PCB (PTQPDPMV1 is default).

4.10.3 Configuration Validation & SETCRC Function Block

The configuration validation functionality prevents the module from causing unexpected results after it receives a new configuration (for example, if it receives a configuration that should have been downloaded to a different module). The PTQ-PDPMV1 module constantly transfers two CRC values to the processor (as part of the input status data) which are calculated based on its current configuration. The processor must copy back the same CRC values to the module (as part of the output status data). If the CRC values are not synchronized, the module will be switched to STOP mode and the CFG ERR LED will be illuminated. If the CFG ERR LED is OFF it means that the CRC values are synchronized. While the module is in STOP mode there will be no data exchange with the configured PROFIBUS slaves.

The SETCRC function block is presented in this procedure for convenience purposes to get your PTQ-PDPMV1 module up and running. The SETCRC function block automatically updates the CRC through the following lines of structured text code:

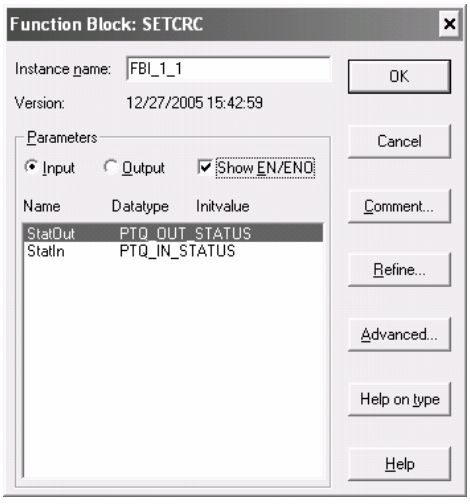
StatOut.PROFIBUSCRC:=StatIn.PROFIBUSCRC;
StatOut.ModuleCRC:=StatIn.ModuleCRC;

This procedure suggests the use of the SETCRC function block in order to avoid the module being set to STOP mode during successive configuration changes (these changes typically occur during the initial setup steps). However, please note that the SETCRC function block also prevents the module from supporting the configuration validation functionality, because the CRC values will always be synchronized even if an unexpected download occurs.

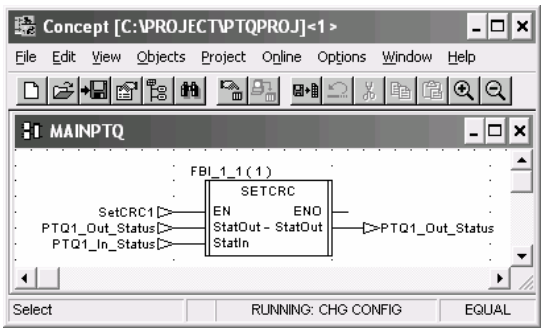
After the module is configured and the CRC values are synchronized, the SETCRC function block can be disabled (for applications that require configuration validation).

To disable the SETCRC function block, follow these steps:

1 Double-click at the SETCRC function block at the FBD section
2 Select the *Show EN/ENO* checkbox



Create a BOOL variable and associate it to the EN input pin of the SETCRC function block. If this variable has a value of 0 (Off) the SETCRC function block will be disabled. Therefore, further changes to the module configuration would cause the module to be switched to STOP mode.



Note: If you use the EN bit to disable the function block, please remember that after a processor download, the output variables might be reset and may cause a CRC mismatch. After a processor download, you may need to re-enable the SETCRC function block once, to synchronize the CRCs again.

4.11 Using Mailbox Function Blocks

4.11.1 Overview

The mailbox function blocks build mailbox requests to the module and read the mailbox response from the module. These mailbox function blocks are optional, meaning that the project will update PROFIBUS data and status information even if no function blocks are used.

4.11.2 Configuration

The mailbox function block contains input, outputs and inputs/outputs pins that must be associated to specific variables.

The mailbox function blocks (except the Get Alarm that will be covered later) require the usage of the following pins (common for all mailbox function blocks):

Pin Name	Pin Type	Description
StatIn	Input	Input Status pin. Must be associated to the imported variable PTQPDPMV1_In_Stat. It contains the status transferred from the module allowing the mailbox function block to receive the acknowledgment that the mailbox request was processed by the module. It is used also to check if a new mailbox response is available. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).

Pin Name	Pin Type	Description
MailIn	Input	Input Mailbox pin. Must be associated to the imported variable PTQPDPMV1_In_Mail. It contains the mailbox response message that is handled by the function block according to its mailbox ID. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
StatOut	Input/Output	Output Status pin. Must be associated to the imported variable PTQPDPMV1_Out_Stat. It is used to check if a new mailbox response is available. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
MailOut	Input/Output	Output Mailbox pin. Must be associated to the imported variable PTQPDPMV1_Out_Mail. This variable stores the mailbox output variable that is updated from the function block when a new mailbox request is performed to the module. It consists on an array of words. **Note:** The actual variable name corresponds with the module name you configured in PCB. The data type names in these examples use the default module name (PTQPDPMV1).
r"Trigger"	Input/Output	Move a value of 1 to this register to initiate the mailbox request. A request can only be initiated if its current value is 0 and all triggers for the other mailbox function blocks also has a value of 0. The actual name for this trigger register will be specific for each mailbox function block. For example, for the SetOperate mailbox (SETOPMD) the trigger registers is *SetOper*.

[1]The actual variable name will depend on the module name configured by the user at PCB. These data type names are considering the default module name (PTQPDPMV1)

4.11.3 Trigger Register

The mailbox requests are initiated by the "trigger" register (INT) that is defined as an input/output pin. The mailbox request is initiated after the application moves a value of 1 to the trigger register.

The current value of all mailbox triggers for you application must be equal to 0 in order to perform a mailbox request. If you are using more than one mailbox function block, you must add a program code to guarantee that this condition is satisfied. Therefore, a mailbox function block is only allowed to send a new mailbox request after the processor receives the confirmation that the previous mailbox request was acknowledged by the module. This condition is determined by checking the status of all trigger registers (0=OK). It also prevents an attempt to send more than one mailbox request in a single PLC scan.

The following table shows the trigger registers used for each mailbox function block:

File Name	Description	Trigger
ACCREAD.ASC	Acyclic Read Mailbox	AcRead
ACCWRITE.ASC	Acyclic Write Mailbox	AcWrite

File Name	Description	Trigger
GETALARM.ASC	Alarm Mailbox	-
GETCFG.ASC	Get Configuration Mailbox	GetCnfg
GETDIAGN.ASC	Get Diagnostics Mailbox	GetDiag
GETLIVE.ASC	Get Live List Mailbox	GetList
SETADDRS.ASC	Set Slave Address Mailbox	SetAddr
SETOPMD.ASC	Set Operating Mode Mailbox	SetOper, SetStop, SetClear
SETSLVMD.ASC	Set Slave Mode Mailbox	SetMdSlv
SETSLVS.ASC	Start/Stop Slaves Dynamically	Start, Stop
ColdBT.ASC	Remote Coldboot from PLC	Cldboot

The Get Alarm function block does not require a trigger because this mailbox is initiated from the module (as covered later in this User Manual).

The trigger register is a variable that can assume different states as follows:

Value	Description
0	OK to send new mailbox request. The last mailbox request was already acknowledged by the module.
1	Mailbox request to be performed. The Quantum program should make sure that the required conditions are satisfied (as previously discussed) before moving a value of 1 to the trigger register. The function block will then build the mailbox request by copying all mailbox input parameters to the mailbox output variable that is transferred to the module through the backplane. Then the function block will automatically change the trigger's value to 2.
2	Processor has performed the mailbox request and is waiting for the acknowledgment from the module. The acknowledgment informs that the module has received the request (the actual mailbox response is actually sent later). After the acknowledgment is received the function block will reset the trigger's value back to 0.

4.11.4 Specific Input Pins

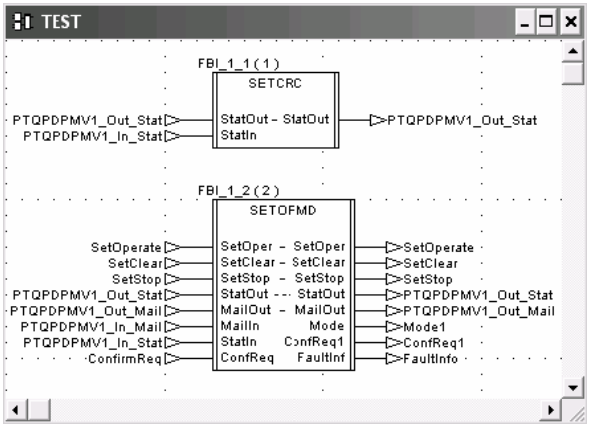
Each function block has input pins specifically for each mailbox. For example, in order to send a Get Diagnostics mailbox, the application must set the PROFIBUS slave address input pin. The processor program must configure the input pins before performing the mailbox request by moving a value of 1 to the mailbox trigger. For a description of each function block input pin you can double-click at the function block instance and select Advanced… for the comment about each input pin. Also refer to the module documentation for detail information about each mailbox parameter.

4.11.5 Specific Output Pins

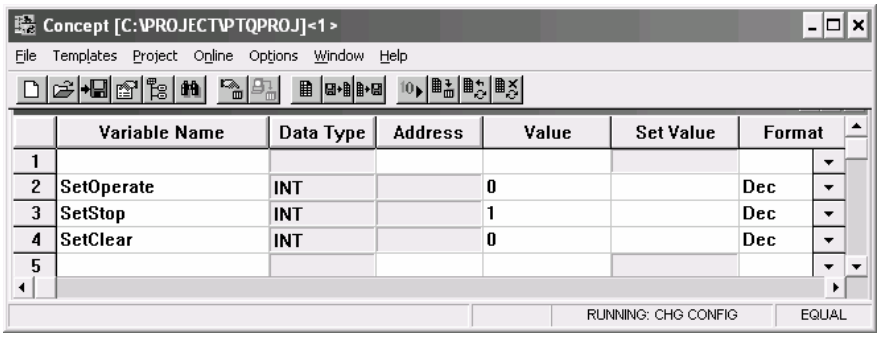
Each function block contains output pins that are updated after the mailbox response is received by the processor. For example, the Get Diagnostics function block has an *ExtDiag* output pin that stores the diagnostic information received from the slave.

Example

If the Set Operating Mode mailbox function block is used as follows:



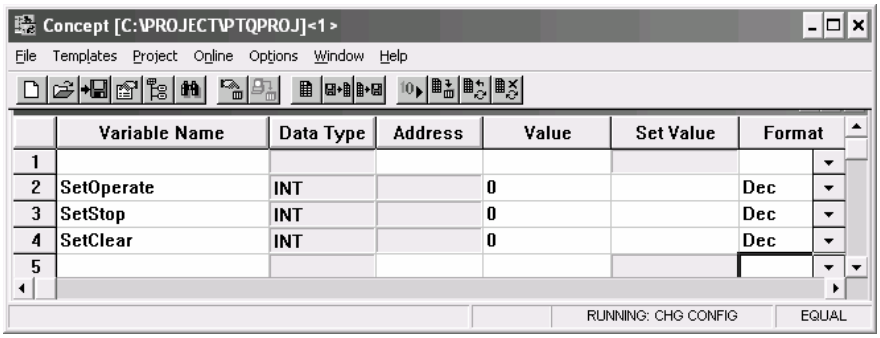
You can create a Reference Data Editor table with three trigger variables for the Set Operating Mode mailbox. Start by moving a value of 1 to SetStop in order to set the module's mode to STOP.



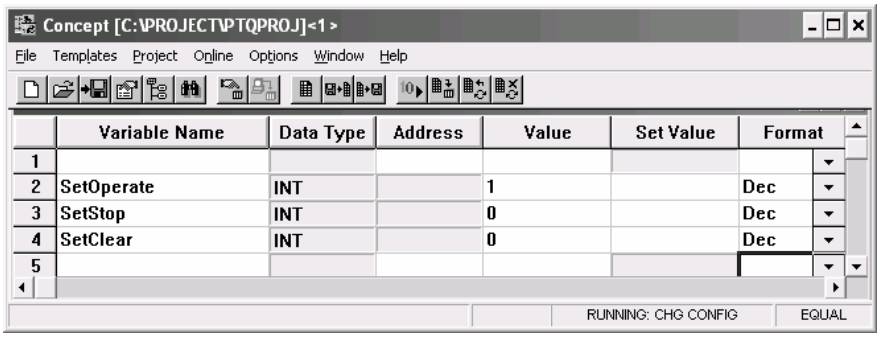
At this point you should notice the following LED display, indicating that the module's mode was changed to STOP:

LED	Status
MSTR STAT	RED
COM STAT	OFF
DBASE STAT	GREEN
TK HOLD	GREEN

You will also notice that the function block automatically clears the trigger register after it receives the acknowledgment from the module.


Concept [C:\PROJECT\PTQPROJ]<1 >
File Templates Project Online Options Window Help
Variable Name | Data Type | Address | Value | Set Value | Format
1
2 SetOperate | INT | 0 | Dec
3 SetStop | INT | 0 | Dec
4 SetClear | INT | 0 | Dec
5
RUNNING: CHG CONFIG
EQUAL

Move a value of 1 to the SetOperate trigger variable.

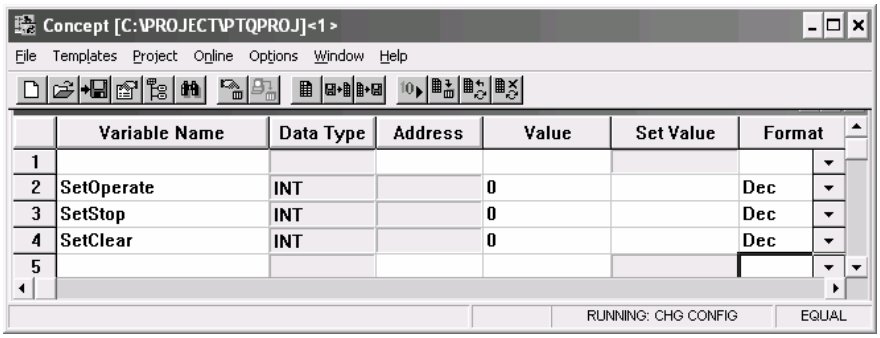

Concept [C:\PROJECT\PTQPROJ]<1 >
File Templates Project Online Options Window Help
Variable Name | Data Type | Address | Value | Set Value | Format
1
2 SetOperate | INT | 1 | Dec
3 SetStop | INT | 0 | Dec
4 SetClear | INT | 0 | Dec
5
RUNNING: CHG CONFIG
EQUAL

At this point you should notice the following LED display, indicating that the module's mode was changed to OPERATE:

LED	Status
MSTR STAT	GREEN
COM STAT	GREEN or OFF[1]
DBASE STAT	GREEN
TK HOLD	GREEN

[1]The COM STAT LED will be either GREEN if the master is communicating with all slaves, blinking if it is communicating with some of the slaves or OFF if it is not communicating with any slaves.

You will also notice that the function block automatically clears the trigger register after it receives the acknowledgment from the module.



4.12 Mailbox Overview

This section provides a brief description on how to use each mailbox function block. Refer to ***Mailbox Messaging*** (page 112) for detailed information about each mailbox parameter:

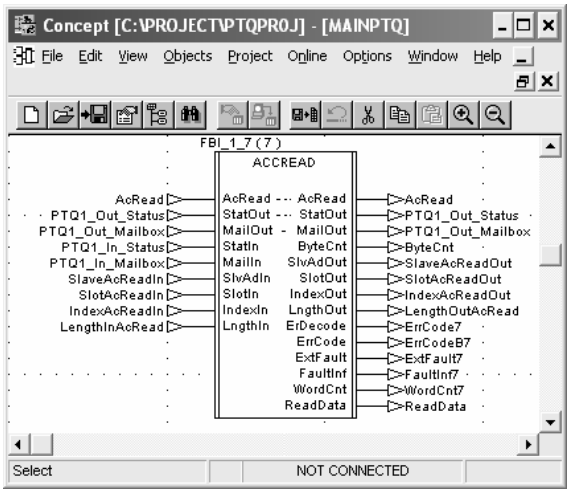
4.12.1 Acyclic Read Mailbox

Function Block: ACCREAD

Trigger Register: AcRead

Description: The ACCREAD mailbox is used to perform an Acyclic Read request to a PROFIBUS slave device. The input pins SlvAdIn (PROFIBUS slave address), SlotIn (slot number), IndexIn (index number) and LngthIn (length - number of bytes associated to acyclic read operation) must be configured before triggering the mailbox request. The acyclic read response data is copied to the ReadData output pin. The status information is available in the output pins (ErrCode, ErrDecode, ExtFault and FaultInf).

The following illustration shows a sample instance of the Acyclic Read mailbox function block:



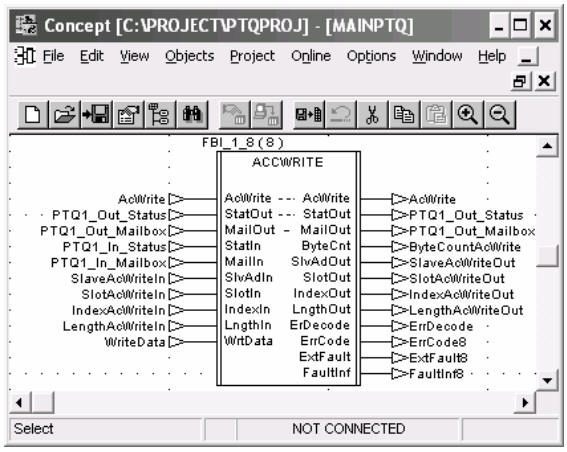
4.12.2 Acyclic Write Mailbox

Function Block: ACCWRITE

Trigger Register: AcWrite

Description: The ACCWRITE function block is used to perform an Acyclic Write request to a PROFIBUS slave device. The input pins SlvAdIn (PROFIBUS slave address), SlotIn (slot number), IndexIn (index number) and LngthIn (length - number of bytes associated to acyclic read operation) must be configured before triggering the mailbox request. The actual data to be written to the PROFIBUS slave should be associated to the WrtData input pin. The status information is available at the output pins (ErrCode, ErrDecode, ExtFault and FaultInf).

The following illustration shows a sample instance of the Acyclic Write mailbox function block:



4.12.3 Alarm Mailbox

Function Block: GETALARM

Trigger Register: -

Description: The GETALARM mailbox is used to read the alarm mailbox messages sent by the module. The module will automatically generate the alarm mailboxes after it receives the alarm message from the PROFIBUS slave. Therefore no triggers are required for this mailbox.

The last alarm received is copied at the *LastAlarm* output pin. This is a data structure that contains all alarm information:

```
ALARMTYPE definition:
       SlaveAddress : BYTE;
       SlotNumber : BYTE;
       SeqNumber : BYTE;
       SpecAck : BYTE;
       AlarmType : BYTE;
       ExtDiag : BYTE;
       FaultInfo : ARRAY[0..1] OF BYTE;
       ByteCount : BYTE;
       Data: ARRAY[0..127] OF WORD;
```

This function block also keeps track of the last 20 alarms through the HistAlarm output pin.

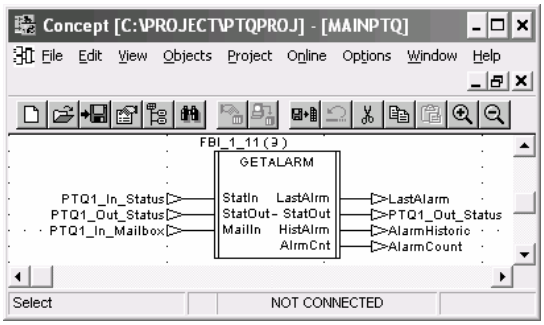
For example, let us suppose that the module receives 10 alarms (first alarm - Sequence Number = 1, second alarm - Sequence Number = 2, and so on). So after the alarm #10 is received the processor application could refer to these alarms stored at the following output pins:

```
Last Alarm - Alarm #10
HistAlarm[1] - Alarm # 9
HistAlarm[2] - Alarm # 8
HistAlarm[3] - Alarm # 7
HistAlarm[4] - Alarm # 6
HistAlarm[5] - Alarm # 5
HistAlarm[6] - Alarm # 4
HistAlarm[7] - Alarm # 3
HistAlarm[8] - Alarm # 3
HistAlarm[9] - Alarm # 2
HistAlarm[10] - Alarm # 1
```

If the HistAlarm buffer is full and it receives a new alarm then the oldest alarm in the queue will be deleted to reserve space for the new alarm.

The AlarmCnt output pin in incremented every time the alarm mailbox is received. This register will rollover at 30000. The processor application can keep track of this register to determine when the processor has received a new alarm mailbox message from the module.

The following illustration shows a sample instance of the GetAlarm mailbox function block:

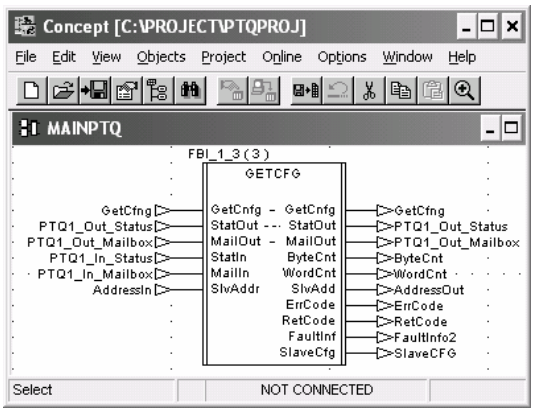


4.12.4 GetConfiguration Mailbox

Function Block: GETCFG

Trigger Register: GetCnfg

Description: The GETCFG function block can be used to read the configuration of any PROFIBUS slave connected to the PTQ-PDPMV1 module. The SlvAddr input pin must be configured with the PROFIBUS slave address of the PROFIBUS device. The configuration data is stored at the SlaveCfg output pin. The byte count of the slave configuration is stored at ByteCnt output pin. The ErrCode, RetCode and FaultInf output pins can be used for status verification.



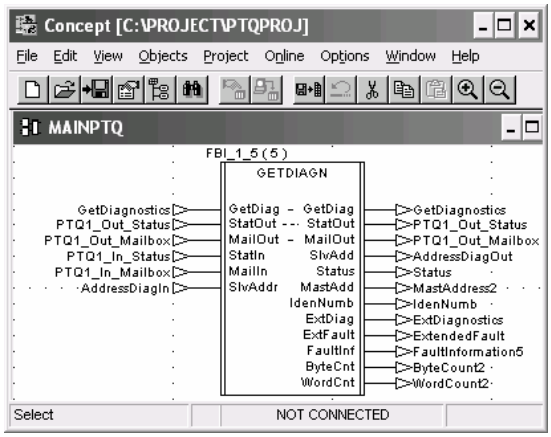
4.12.5 GetDiagnostics Mailbox

Function Block: GETDIAGN

Trigger Register: GetDiag

Description: The GETDIAG function block can be used to read the diagnostics from any PROFIBUS slave connected to the PTQ-PDPMV1 module. The slave address must be set at the SlvAddr input pin. The diagnostics data is copied at the ExtDiag output pin. The number of bytes of the diagnostics message is

stored at the ByteCnt output pin (status+identification+extended diagnostics). The ExtFault and FaultInf output pins can be used for status information. The master address is stored at the MastAdd output pin.



4.12.6 GetLiveList Mailbox

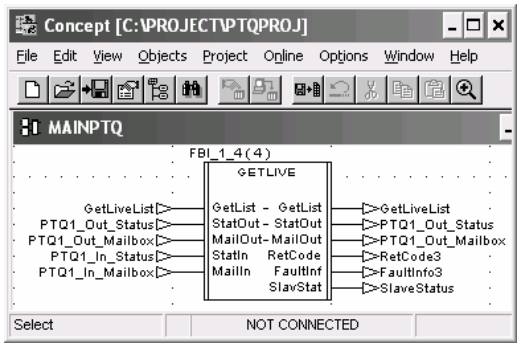
Function Block: GETLIVE

Trigger Register: GetList

Description: The GETLIVE function block can be used to read the live list from the module containing the status of each device at the PROFIBUS network. The live list is stored at the SlavStat output pin. The live list data is an array of bytes stored as follows:

```
SlaveStat[0] - status of device configured with PROFIBUS address 0
SlaveStat[1] - status of device configured with PROFIBUS address 1
SlaveStat[2] - status of device configured with PROFIBUS address 2
SlaveStat[3] - status of device configured with PROFIBUS address 3
SlaveStat[4] - status of device configured with PROFIBUS address 4
Etc…
```

Refer to the User Manual for more information about the valid status codes. The RetCode and FaulInf output pins can be used for mailbox status information.

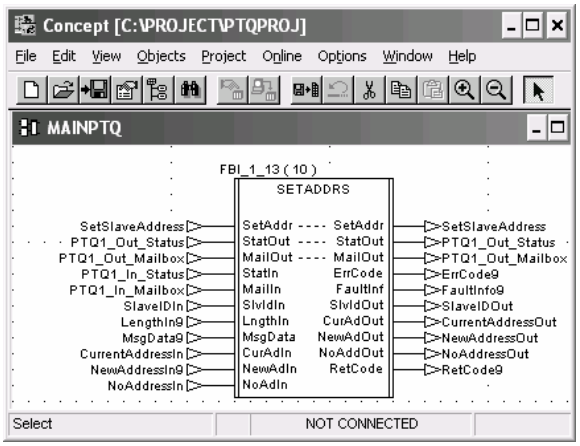


4.12.7 SetSlaveAddress Mailbox

Function Block: SETADDRS

Trigger Register: SetAddr

Description: The SETADDRS mailbox can be used to change the slave address. Only specific PROFIBUS devices support this feature. The application must set the CurAdIn (current address) and NewAdIn (new address) input pins. It is also possible to deliver user data through the MsgData input pin (the number of bytes must be set through the LgnthIn input pin). The SlvIdIn input pin must be set with the Ident number for the slave. The FaultInf output pin can be check for mailbox status information.



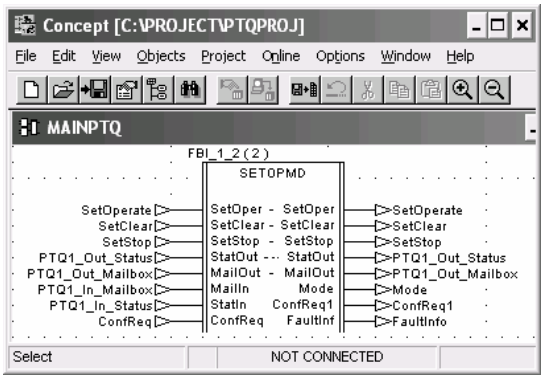
4.12.8 SetOperatingMode Mailbox

Function Block: SETOPMD

Trigger Register: SetOper, SetStop, SetClear

Description: The following trigger values can be used to change the current operating mode of the module:

```
SetOper = Set Operate
SetStop = Set Stop
SetClear = Set Clear
```

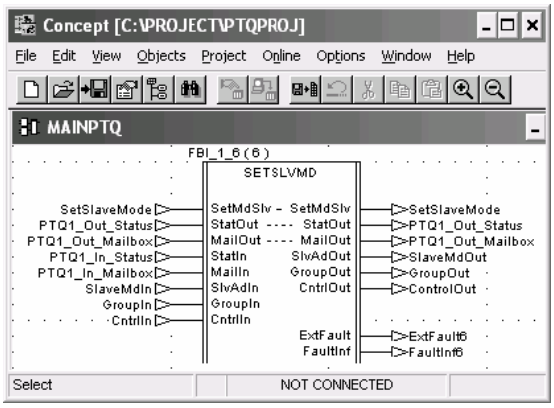


4.12.9 SetSlaveMode Mailbox

Function Block: SETSLVMD

Trigger Register: SetMdSlv

Description: The SETSLVMD function block can be used to request the module to sync, unsync, freeze or unfreeze. The slave address must be selected through the SlvAddIn input pin. If the operation is directed to a group of slaves then the Group number must be set through the GroupIn input pin parameter. The actual code that will select the operation type must be configured through the CntrlIn input pin. Refer to the User Manual for the valid control codes.

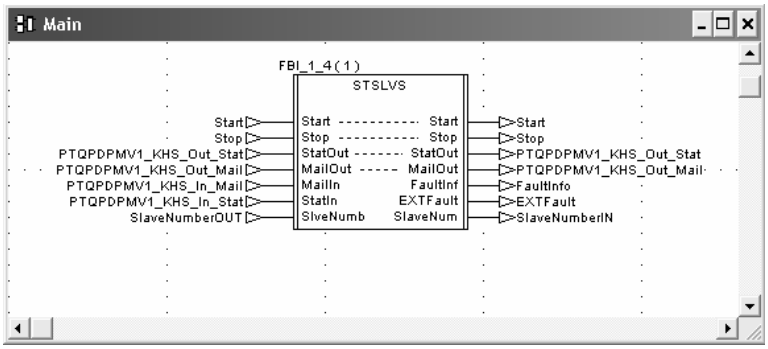


4.12.10 Start/Stop Slaves Mailbox

Function Block: STSLVS

Trigger Byte: Start / Stop

Description: The STSLVS function block can be used to request the module to Start/stop certain slaves dynamically. The slave address must be selected through the SlveNumb input pin. This is an array of 126 slaves. Change the value for a specific slave from 0 to 1 to stop communication with the master



The following illustration shows that when you execute the mailbox command, Slave #4 and Slave #13 will Start/stop communicating with the master.

RDE Template (NEWTEST.RDE) - Animation OFF

	Variable Name	Data Type	Address	Value	Set Value	Format
1	Start	BYTE		0	0	Dec
2	Stop	BYTE		0	0	Dec
3						
4	SlaveNumberOUT[4]	BYTE		1	0	Dec
5	SlaveNumberOUT[13]	BYTE		1	0	Dec
6						
7						

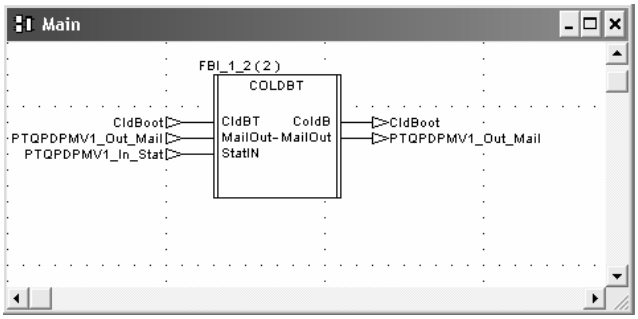
4.12.11 Coldboot Mailbox

Important: The Coldboot mailbox is only supported on PTQ-PDPMV1 modules running firmware version 1.19 or newer. Earlier versions of the firmware do not support this feature. If you require this functionality, please contact ProSoft Technical Services for information on how to upgrade your module.

Function Block: COLDBOOT

Trigger Byte: ColdBoot

Description: The COLDBOOT function block allows you to remotely reboot the module. To trigger a reboot the module, change the value of the Cldboot bit from OFF (zero) to ON (one). The bit is reset back to OFF when the function is executed.



RDE Template (TEST.RDE) - Animation OFF

	Variable Name	Data Type	Address	Value	Set Value	Format	Disable	Cyclic Set
1	CldBoot.Coldboot	BOOL		Off		Bool		
2								
3								

HSBY Note: This function block will reset both the active master and the passive master.

Using a Control Word to Reboot the Module

If you need to coldboot the module without using the Coldboot mailbox, use the PTQPDPMV1_OUT_Stat.LastinMailID [0] control word.

Note: Even though Mailbox is not being used but the name (PTQPDPMV1_OUT_Stat.LastinMailID [0]) still being used due to fact that in DTY file, name can't be changed dynamically for Concept Application user need to refer to the user manual references for more details.

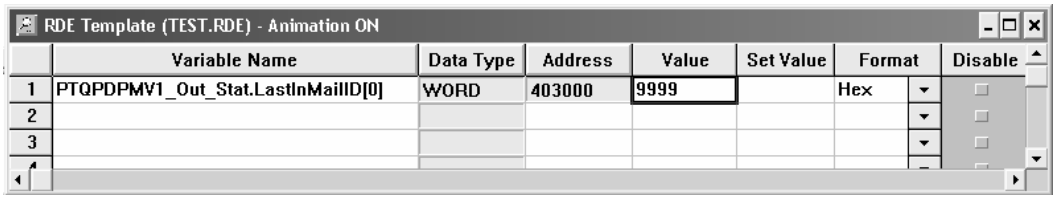
When mailbox is used word zero in the output image is used for mailbox handshaking (PTQPDPMV1_OUT_Stat.LastinMailID [0])

When mailbox is not used then word zero in the output image is for control of the operating mode and cold boot and supposed to be named (PTQPDPMV1_OUT_Stat.Setoperatingmode[0]).

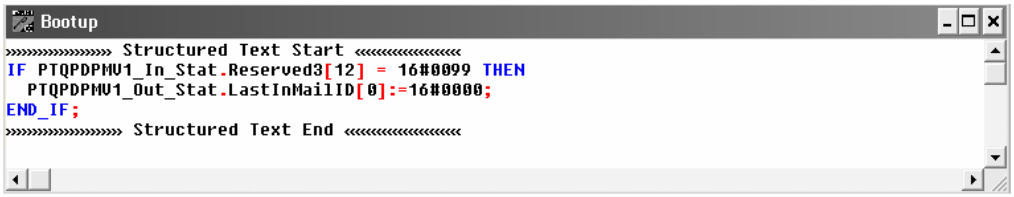
Remember only name stays the same but operation will work as it supposed to.

➢ ***To reboot the module:***

1 Enter the value 16#9999 (Hex) in the PTQPDPMV1_Out_Stat.LastInMailID[0] register, as shown in the following illustration. For this application, word zero is used for control and not for mailbox handshaking.



2 Add the following Structured Text lines to the program file:



In order to reset the logic, the program must be written in a way to look for word 72 in the input status. If word 72 is equal to 16#0099 hex, the logic will reset the value in PTQPDPMV1_Out_Stat.LastInMailID[0] to 16#0000 (Hex).

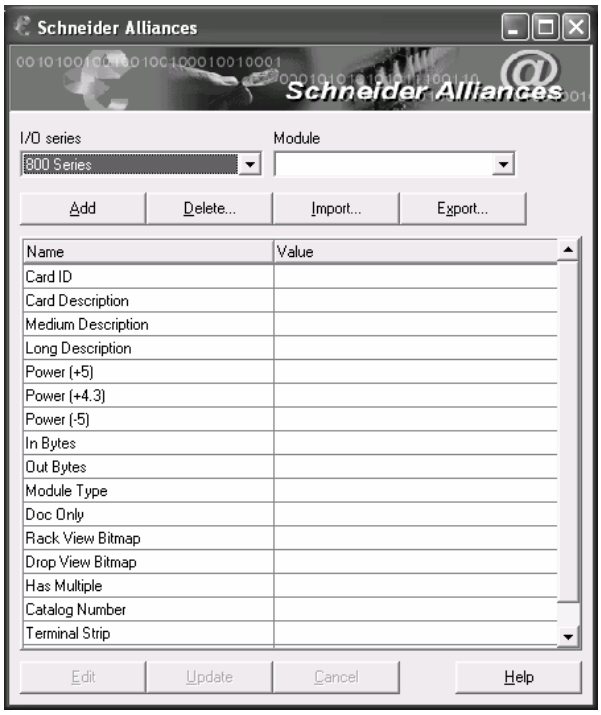
Note: It is normal for the Passive Master in Hot Standby applications to reboot twice using this procedure.

5 Configuring the Processor with ProWORX 32

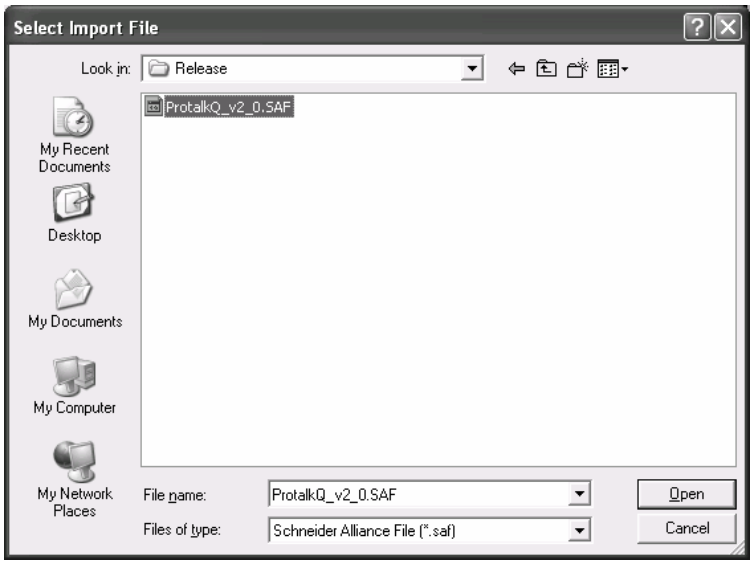
HSBY Note: ProWorx32 software does not support 140CPU67160 processor and therefore does not support the PTQ-PDPMV1 HSBY functions.

1 Run the Schneider_Alliances.exe application that is installed with the Proworx 32 software:

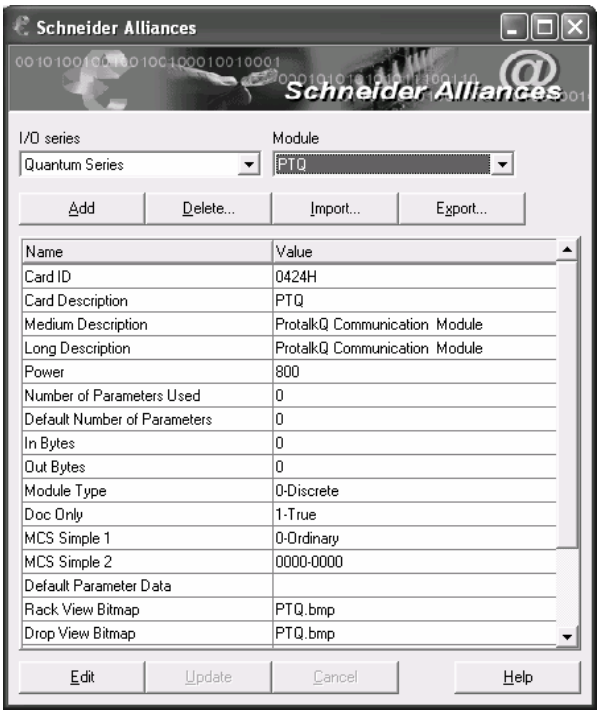
2 Click on Import…



3 Select the .SAF File that is located at the CD-ROM shipped with the PTQ module



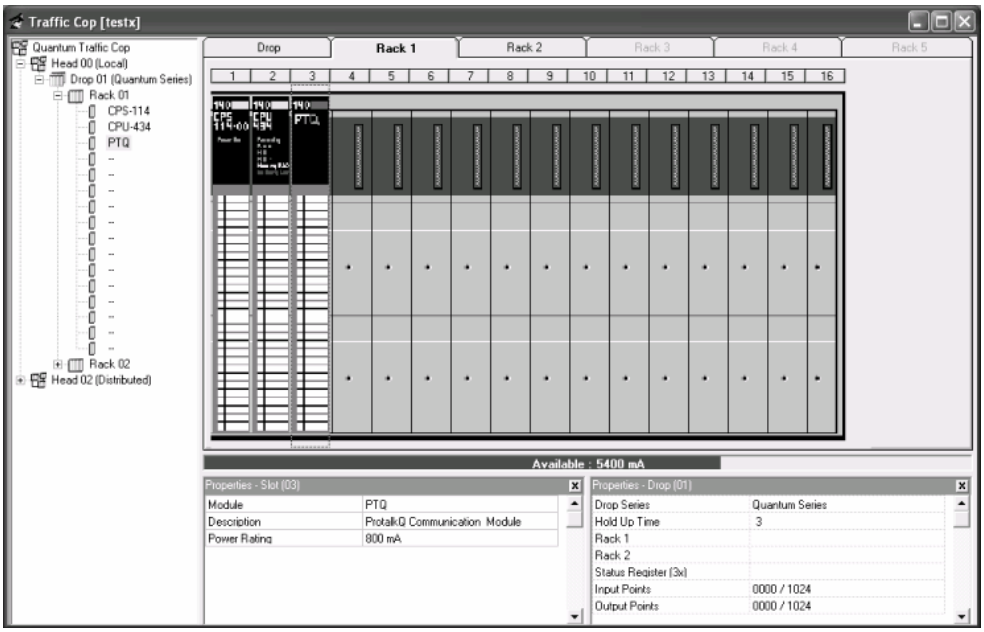
4 Select Open to import the PTQ module profiles (select I/O series as Quantum):



The following modules will be available after importing the .SAF file:

Card Description	Use With Following Module(s)
PTQPDPMV1	PTQ-PDPMV1 module
PTQ	All PTQ modules except PTQ-PDPMV1

5 Close the Schneider Alliances application and run the Proworx software.
6 At the Traffic Cop section select either the PTQ or PTQPDPMV1 cards to be inserted at the selected slot.



6 Mailbox Messaging

In This Chapter

- Mailbox Message Queuing 113
- Special Function Mailbox Messaging Commands 114
- Receiving Mailbox Message Responses from module 132
- Mailbox Messaging Error Codes 133

The PTQ-PDPMV1 PROFIBUS DP master uses a protocol called "mailbox messaging" to exchange parameter data between the processor, master and slave devices. This protocol provides a way to encapsulate and prioritize commands and data sent between the PROFIBUS Master and slaves.

This communication method performs a key feature of PROFIBUS version DP-V1 called acyclic data communication.

Acyclic communication extends data communication for parameterization and calibration of field devices over the bus during runtime, and for the introduction of confirmed alarm messages. Acyclic data is transmitted in parallel with cyclic data communication, but with lower priory.

Cyclic data commutation is the exchange of actual slave I/O data, and is handled automatically by the master in defined, recurring deterministic sequence.

Mailbox messaging commands are incorporated into the sample ladder logic. Mailbox messages, and responses to mailbox messages, are stored in mailbox data objects.

The following topics describe the structure of each mailbox message object in the sample program.

6.1 Mailbox Message Queuing

The PTQ-PDPMV1 module operates asynchronously on the Mailbox Messages and as such provides for the queuing of the messages as they are received. The queue sizes used in the module are as follows:

Queue Type	Queue Size Max	Description
Output Message from processor	126	Number of messages that the PTQ module will queue by type of message. Note that status of the queues can be monitored via the Queue Message Count values.
Input Messages for processor	126	
Alarm Messages from Slaves for processor	100	

6.1.1 Queue Timeouts

The PTQ-PDPMV1 module will only allow a message to stay in a queue for up to 10 seconds. If the PROFIBUS Master (for Output Messages) or the processor (for Input and Alarm messages) have not successfully received a message within 10 seconds, the module will clear the message out of the queue.

6.2 Special Function Mailbox Messaging Commands

The PTQ-PDPMV1 module supports some extended PROFIBUS functions, using a mailbox data exchange mechanism implemented in the PTQ-PDPMV1 module.

The PTQ-PDPMV1 module supports the following special functions through this mailbox messaging scheme:

Initiated from Processor

Message	Description
Set Operation Mode	Controls the operating state of the PROFIBUS Master
Set Slave Mode	Send special control command to one or several slaves (Sync/Freeze)
Get Slave Diag	Get diagnostic information from a slave
Get Slave Config	Get slave configuration
Set Slave Address	Set node address of a slave (if supported by the slave)
Get Live List	Get information from all nodes on the network
MSAC1 Read	DPV1 acyclic read (Class 1)
MSAC1 Write	DPV1 acyclic write (Class 1)

DPV1 Alarm Handling - Generated by Slave Devices

Message	Description
Alarm Indications	Spontaneous Alarm indication from DPV1 slave. Structure of data is slave dependent
Alarm Confirmation	This message is sent **by the PTQ module** automatically as a confirmation to the Alarm Indications.

The processor logic required to implement these messaging mailbox exchanges will be made simpler after the Function Blocks are provided.

Sending a mailbox message to the PTQ-PDPMV1 module is a relatively simple process, however, it is important to follow a certain sequence.

Remembering the PROFIBUS Output Data Memory Map:

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
40101	%MW101	0	Last in Mailbox Message ID
40102	%MW102	1	Last Alarm Control index
40103	%MW103	2	PROFIBUS CRC32 - Computed for PROFIBUS Config
40104	%MW104		

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
40105	%MW105	4	Module CRC32 - computed for module data
40106	%MW106		When the module first starts up or recognized an initialization of the processor, it will compare the values of the two CRCs in the input and output images. If either one of the CRCs do not match, the module will be placed in Stop mode. If each set matches, the module will be placed in Operate mode.
40107 to 40150	%MW107 to %MW150	6 to 149	Outgoing Mailbox Data - Mailbox Message command being sent to the PTQ module
40151 to N	%MW151 to N	150 to N	PROFIBUS Output Data - Data going to the PROFIBUS Network N is a function of the user selected size of the PROFIBUS Output data block. Maximum size is 1536 bytes

The important section relevant to the Mailbox Messaging discussion is the Outgoing Mailbox Data section (Word Offsets 2 to 145). Within this section of data, the following structure exists:

Mailbox Message Structure - To PTQ module

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Type	Description
40107	%MW107	6	Message ID	Processor logic or user set. The Message ID field is used by the PTQ module to detect a new message in the PROFIBUS Output data image. When the value is detected as non-zero, the message is processed immediately.
40108	%MW108	7	Message Info	See individual commands for data values to be entered in each of these register locations
40109	%MW109	8	Command	
40110	%MW110	9	Data Size	
40111	%MW111	10	Frame Count	
40112	%MW112	11	Frame Number	
40113	%MW113	12	Offset high	
40114	%MW114	13	Offset Low	
40115	%MW115	14	Extended Word 1	
40116	%MW116	15	Extended Word 2	
40117	%MW117	16	Extended Word 3	
40118	%MW118	17	Extended Word 4	

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Type	Description
40119	%MW119	18	Extended Word 5	
40120	%MW120	19	Extended Word 6	
40121	%MW121	20	Extended Word 7	
40122	%MW122	21	Extended Word 8	
-		-	See individual commands	
40150	%MW150	149		

Please keep the following key points in mind:

- If no message is to be sent in the mailbox, the Message ID value in the Output Image must be set to zero (0).
- The PTQ module will set the Last Out Mailbox Message ID value in the Input Image to zero (0).
- The Message ID field is used by the PTQ module to detect new outgoing messages. When the value is detected as non-zero, the PTQ processes the message immediately.
- The Message ID should be changed for each new outgoing mailbox message. A simple zero to one toggle scheme can be used, or an incrementing register value can be used (1 to 15).
- When a new message is to be sent:
  1. Copy or setup the message contents (keep Message ID value at zero) to the Output Data locations.
  2. Set the Message ID value to PTQ module.
- After the PTQ module processes the Outgoing Mailbox command, the PTQ will set the Last_Out_Mailbox_Message_ID in the Output Data image to match the outgoing Message ID in the Output image. This event on the processor side can be used by the processor logic to clear the outgoing Message ID if desired.

6.2.1 Mailbox Message - Set Slave Mode

In addition to station related user data transfer, which is executed automatically, the master can send control commands to a single slave, a group of slaves, or all slaves simultaneously. These control commands are transmitted as multicast commands. This permits use of sync and freeze modes for event controlled synchronization of the slaves.

The slaves begin sync mode when they receive a sync command from their assigned master. The outputs of all addressed slaves are then frozen in their current state. During subsequent user data transmissions, the output data are stored at the slaves, but the output states remain unchanged. The stored output data are not sent to the outputs until the next sync command is received. Sync mode is concluded with the unsync command.

Similarly, a freeze control command causes the addressed slaves to assume freeze mode. In this operating mode, the states of the inputs are frozen until the master sends the next freeze command. Freeze mode is concluded with the unfreeze command.

Note 1: It is only possible to send control commands when operating mode is either "CLEAR" or "OPERATE".

Note 2: Not all slaves support this feature. Refer to the documentation for the actual slave for more information.

Parameter	Description
Command Initiator	Application
Command Name	SET SLAVE MODE
Command Number	0300h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout - Set Slave Mode

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0240h		0200h		
Command	0300h		0300h		Set Slave Mode
Data size	0000h		0000h		
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Group Select	Slave Address	Group Select	Slave Address	
Extended word 2		Control Command		Control Command	
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		-		
Extended word 6	-		-		
Extended word 7	-		Extended Fault Information		

Message Information

Refer to ***Message Information*** (page 134).

Slave Address

Range 1 to 125; 127

If the request applies for only one slave, that Slave Address must be entered in the range 1 to 125. If a slave group is to be addressed, Slave Address should be

127 (Multicast address). The value entered will be byte swapped. Example a slave address of 0014 would be entered as 0E00h

Group Select

Range 01h to FFh (Bit Coded)

This parameter decides which group should be addressed. Refer to the following example:

b7	b6	b5	b4	b3	b2	b1	b0
Group 4	Group 3	Group 2	Group 1	Group 8	Group 7	Group 6	Group 5

Example: To address Group 1, 2, and 4, the Group Select value should be D0h. If an individual slave should be addressed, the correct group selection must also be made as the slave will ignore the message if it does not belong to the requested group(s).

What group(s) a slave belongs to is determined during network configuration, and is downloaded during initialization to each slave via the PROFIBUS telegram "Set_Prm".

Control Command

This parameter specifies the command to send:

Bit	Explanation
0 (LSB)	Reserved, set to zero
1	Reserved, set to zero
2	Unfreeze input data
3	Freeze input data
4	Unsynchronize output data
5	Synchronize output data
6	Reserved, set to zero
7 (MSB)	Reserved, set to zero

Combinations of the bits (Unsync/Sync and Unfreeze/Freeze

Bits 0 or 6	Bits 1 or 7	Explanation
0	0	No Function
0	1	Function will be activated
1	0	Function will be inactive
1	1	Function will be inactive

Fault Information and Extended Fault Information

"Fault Information" Contents		"Extended Fault Information" Contents	
0100h	Address out of range	-	
0200h	Group number 0 not permitted	-	
0A00h	Failed to send Global Control request	0A00h	Incorrect operation mode (Clear/Operate Only)
		0150h	Invalid Freeze Group (Group is not initiated to be Freeze Group)
		0250h	Invalid Sync Group (Group is not initiated to be Sync Group)

"Fault Information" Contents		"Extended Fault Information" Contents	
		0350h	Incorrect Control Command
		0450h	No Sync -/ or Freeze groups enabled in master configuration
FE00h	Command not possible in Class 2 only mode	-	
FF00h	Module not initialized	-	

6.2.2 Mailbox Message - Get Slave Diagnostics

This command reads diagnostic data from a specified slave.

Note: The response data size depends on the actual slave implementation. Range 6 to 244.

Parameter	Description
Command Initiator	Application
Command Name	GET SLAVE DIAGNOSTICS
Command Number	0400h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout - Get Slave Diagnostics

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0240h		0200h		
Command	0400h		0400h		Get Slave Diagnostics
Data size	0000h		(Size of data)		
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Type of request	Slave Address	Type of request	Slave Address	
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		Error code 2	Error code 1	
Extended word 6	-		Error code 4	Error code 3	
Extended word 7	-		Return Code		
Extended word 8	-		Fault Information		
			Station Status 2	Station Status 1	Response data word 1

Command

Response		
Station Status 4	Station Status 3	Response data word 2
Ident Number		Response data word 3
Extended Diagnostic Data		Response data word 4 ... ... Response data word n

Message Information

Refer to ***Message Information*** (page 134).

Slave Address

Range 1 to 125, specifies the slave to read diagnostics from.

Type of request

00h: Internal slave diagnostic request. The diagnostic information stored in the master is returned. Can only be requested for slaves configured by the master.

Note: Not allowed when operating in "Class 2-Only" mode.

01h: External slave diagnostic request. A diagnostic request is sent on the network to the specified slave. Can be requested for all slaves on the network.

Error code [1 ...4]

If "Return Code" equals 8030h ("Negative indication from lower layer"), status values according to the DP-specification may be available in "Error Code 1". Error Codes 2 to 4 are reserved.

Refer to ***Mailbox Messaging Error Codes*** (page 133).

Return Code

Refer to ***Mailbox Messaging Error Codes*** (page 133)

Fault Information

If "Invalid Other" is returned in the Message Information word in the header of the response, information about the fault can be found here.

0100h: Address out of range.

0200h: Incorrect "Type of request"

0A00h: Failed to read diagnostic data from slave. Refer to ***Return Codes*** (page 134) for additional fault information.

0B00h: Remote station failure. Refer to ***Return Codes*** (page 134) for additional fault information.

FE00h: Command not possible; module operates as a Class 2 master only.

FF00h: Module offline (not initialized or no valid database).

Station Status [1 ... 3]

Refer to EN50170 Vol. 2 for more information.

Master Address

Address of the master that parameterized the slave.

Ident Number

Unique ID assigned by the PROFIBUS User Organization.

Extended Diagnostic Data

Slave user-specific data. Refer to the documentation for the actual slave for more information.

6.2.3 Mailbox Message - Get Slave Configuration

This command reads the actual configuration (identifier bytes) of a specified slave.

Note: The response data size depends on the actual slave implementation. Range 6 to 244.

Parameter	Description
Command Initiator	Application
Command Name	GET SLAVE CONFIGURATION
Command Number	0500h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout - Get Slave Configuration

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0204h		0200h		
Command	0500h		0500h		Get Slave Configuration
Data size	0000h		(Size of data)		Number of identifier bytes (n)
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1		Slave Address		Slave Address	
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		Error Code 2	Error Code 1	
Extended word 6	-		Error Code 4	Error Code 3	

	Command
Extended word 7	-
Extended word 8	-

Response	
Return Code	
Fault Information	
Identifier byte 1	Response data byte 1
Identifier byte 2	Response data byte 2
Identifier byte 3	Response data byte 3
...	...
Identifier byte n	Response data byte n

Message Information

Refer to ***Message Information*** (page 134).

Slave Address

Range 1 to 125, specifies the slave to read the configuration from.

Error Code [1 … 4]

If "Return Code" equals 3080h ("Negative indication from lower layer"), status values according to the DP-specification may be available in "Error Code 1", Error Codes 2 through 3 are reserved. Refer to ***Mailbox Messaging Error Codes*** (page 133).

Return Code

Refer to ***Mailbox Messaging Error Codes*** (page 133).

Fault Information

If "Invalid other" is returned in the Message Information word in the header of the response, information about the fault can be found here. Refer to ***Message Information*** (page 134).

0100h: Address out of range.

0A00h: Failed to execute request. Refer to ***Return Codes*** (page 134) for additional information.

0B00h: Remote station failure. Refer to ***Return Codes*** (page 134) for additional information.

FF00h: Module not initialized.

Identifier Bytes [1 … n]

Refer to EN50170 Vol. 2 for information on the structure of these bytes. In addition, refer to the documentation provided with the slave device for more information.

6.2.4 Mailbox Message - Set Slave Address

This command makes it possible to set the node address of a specified slave, if the slave supports this feature.

Note: The message data size depends on the actual slave implementation, range 0 to 240 bytes.

Parameter	Description
Command Initiator	Application
Command Name	SET SLAVE ADDRESS
Command Number	0600h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout - Set Slave Address

	Command			Response			
Message ID	(ID)			(ID)			
Message information	0204h			0200h			
Command	0600h			0600h			Set Slave Address
Data size	(Size of data)			(Size of data)			No. of Slave Data bytes (n)
Frame count	0100h			0100h			
Frame number	0100h			0100h			
Offset high	0000h			0000h			
Offset low	0000h			0000h			
Extended word 1	New Slave Address	Current Slave Addr.		New Slave Address	Current Slave Addr.		
Extended word 2	Slave Ident Number			Slave Ident Number			
Extended word 3		No_add_ Chg	-		No_add_ Chg		
Extended word 4	-			-			
Extended word 5	-			Error Code 2	Error Code 1		
Extended word 6	-			Error Code 4	Error Code 3		
Extended word 7	-			Return Code			
Extended word 8				Fault Information			
Message Data byte 1	Slave Data 1			Slave Data 1			(slave data will appear byte swapped)
Message Data byte 2	Slave Data 2			Slave Data 2			
Message Data byte 3	Slave Data 3			Slave Data 3			
...	...			...			
Message Data byte 'n'	Slave Data n			Slave Data n			

Message Information

Refer to ***Message Information*** (page 134).

Current Slave Address

Range 1 to 125, specifies the current address of the slave.

New Slave Address

Range 1 to 125, specifies the new address of the slave.

Slave Ident Number

Ident number for the slave, which address should be altered.

No_add_Chg

This parameter specifies whether it is allowed to change the slave address again at a later stage. If this is not allowed, it is only possible to change the address with this function after initial reset. After the initial reset, the slave takes the default address of 126.

00h: Change of address is still possible at a later stage.

01h-FFh: Change of address is only possible after the initial address (that is, the default address = 126.

Error Code [1 …4]

If "Return Code" equals 3080h ("Negative indication from lower layer"), status values according to the DP-specification in available in "Error Code 1". Error Codes 2 and 3 are reserved. Refer to ***Return Codes*** (page 134).

Return Code

Refer to ***Return Codes*** (page 134).

Fault Information

If "Invalid Other" is returned in the Message Information word in the header of the response, information about the fault can be found here:

0100h: Current slave address out of range.

0200h: New slave address out of range.

0A00h: Failed to execute request.

0B00h: Remote station failure.

FF00h: Module not initialized.

Refer to ***Mailbox Messaging Error Codes*** (page 133).

Slave Data

With this parameter, it is possible to deliver user-specific data. The data is stored in the slave if possible (that is, EEPROM, FLASH, and so on).

6.2.5 Mailbox Message - Get Live List

This command returns 127 bytes of information about the nodes on the network. Every byte stands for one bus subscriber, and the position of the byte in the response data assigns the address (0 to 126), the content assigns the Station Type.

This command can be sent in all operation modes (that is, STOP, CLEAR, and OPERATE), however the module must be initialized properly.

Parameter	Description
Command Initiator	Application
Command Name	GET LIVE LIST
Command Number	1800h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout - Get Live List

	Command	Response	
Message ID	(ID)	(ID)	
Message information	0240h	0200h	
Command	1800h	1800h	Get Live List
Data size	0000h	7F00h	127 Bytes of Data
Frame count	0100h	0100h	
Frame number	0100h	0100h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	Return Code	
Extended word 8	-	Fault Information	
Message Data byte 1		Station Type 0	Response Data Byte 1
Message Data byte 2		Station Type 1	Response Data Byte 1
Message Data byte 3		Station Type 2	Response Data Byte 1
...		...	Response Data Byte 1
Message Data byte 'n'		Station Type 126	Response Data Byte 1

Message Information

Refer to ***Message Information*** (page 134).

Station Type [0 … 126]

00h: Slave Station

01h: Master Station not yet ready for Token ring (station only physically at the bus)

02h: Master Station ready to enter Token ring (there is not yet any Token transmission)

03h: Master Station in Token Ring (Token transmission through the station)

04h: Station does not exist

Fault Information

If "Invalid Other" is returned in the Message Information word in the header of the response, information about the fault can be found here. Refer to ***Message Information*** (page 134).

0AH00: Failed to build Live List.

FF00h: Module offline (not initialized or no valid database)

6.2.6 Mailbox Message - Acyclic Data Read - Class 1

This command initiates a DPV1 Class 1 acyclic read request. Refer to EN50170 (DPV1) for more information.

Parameter	Description
Command Initiator	Application
Command Name	MSAC1 READ
Command Number	2000h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout - Acyclic Read

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0204h		0200h		
Command	2000h		2000h		Acyclic Read
Data size	0000h		(Size of data)		Number of data bytes (n)
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Slot	Slave Addr.	Slot	Slave Addr.	
Extended word 2	Length	Index	Length	Index	
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		Error Decode		
Extended word 6	-		Error Code 2	Error Code 1	
Extended word 7	-		Extended Fault information		
Extended word 8	-		Fault Information		
			Data 1		Response Data byte 1
			Data 2		Response Data byte 1
			Data 3		Response Data byte 1
			...		...
			Data n		Response Data byte 1

Message Information

Refer to ***Message Information*** (page 134).

Slave Address

Station address of the slave responder.

Slot Number and Slot Index

Used in the slave to address the desired data block.

Length

This parameter specifies the number of bytes of the data block that have to be read. If the server data block length is less than requested, the length of the response will be the actual length of the data block. If the server data block is greater or equal, the response will contain the same amount of data.

The slave may answer with an error response if data access is not allowed.

Data [1 … n]

Returned data.

Fault Information and Extended Fault Information

If "Invalid Other" is returned in the Message Information word in the header of the response, information about the fault can be found here.

Fault Information		Extended Fault Information Contents
0100h	Address out of range	-
0A00h	Failed to execute request	Refer to ***Return Codes*** (page 134).
0B00h	Remote station failure	
1000h	Remote station DPV1 failure	Function_Number
1100h	Length out of range (>240 bytes)	-
1200h	Slave does not support DPV1	-
1300h	Slave not active or not present in configuration	-
FE00h	Command not possible in "Class 2-Only" mode	-
FF00h	Module offline (not initialized or no valid database)	-

Error Decode, Error Code 1 and Error Code 2

If "Fault Information" contains error code 1000h, more information according to the DPV1 specification can be found here.

6.2.7 Mailbox Message - Acyclic Data Write - Class 1

This command initiates a DPV1 Class 1 acyclic write request. Refer to EN50170 (DPV1) for more information.

Parameter	Description
Command Initiator	Application
Command Name	MSAC1 WRITE
Command Number	2100h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout - Acyclic Write

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0204h		0200h		
Command	2100h		2100h		Acyclic Write
Data size	(Size of data)		(Size of data)		Number of data bytes (n)
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Slot	Slave Addr.	Slot	Slave Addr.	
Extended word 2	Length	Index	Length	Index	
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		Error Decode-		
Extended word 6	-		Error Code 2	Error Code 1	
Extended word 7	-		Extended Fault information		
Extended word 8	-		Fault Information		
Message Data byte 1	Data 1		Data 1		
Message Data byte 2	Data 2		Data 2		
Message Data byte 3	Data 3		Data 3		
...	...		...		
Message Data byte n	Data n		Data n		

Message Information

Refer to ***Message Information*** (page 134).

Slave Address

Station address of the slave responder.

Slot Number and Slot Index

Used in the slave to address the desired data block.

Length

This parameter specifies the number of bytes that have to be written. If the destination data block size is less than requested, the response will contain an error message. If the data block length is greater than or equal to the requested length, the response contains the number of bytes that have been written. The slave may answer with an error response if data access is not allowed.

Data [1 … n]

Data that should be written.

Fault Information and Extended Fault Information

If "Invalid Other" is returned in the Message Information word in the header of the response, information about the fault can be found here:

Fault Information		Extended Fault Information Contents
0100h	Address out of range	-
0A00h	Failed to execute request	Refer to ***Return Codes*** (page 134).
0B00h	Remote station failure	
1000h	Remote station DPV1 failure	Function_Number
1100h	Length out of range (>240 bytes)	-
1200h	Slave does not support DPV1	-
1300h	Slave not active or not present in configuration	-
FE00h	Command not possible in "Class 2-Only" mode	-
FF00h	Module offline (not initialized or no valid database)	-

Error Decode, Error Code 1, and Error Code 2

If "Fault Information" contains error code 1000h, more information according to the DPV1 specification can be found here.

6.2.8 Mailbox Message - Alarm Indication

This message indicates to the application that a DPV1 slave has transferred an Alarm message to the master. This mailbox message is sent spontaneously by the module. That is, the module itself initiates the mailbox communications.

Detailed information about the cause of the alarm is presented in extended words 1 to 3 and in the message data field (see below).

Command and Response Layout - Alarm Indication

	Command		Response	
Message ID	(ID)		(ID)	Alarm Indication
Message information	0240h		0200h	
Command	2200h		2200h	
Data size	(request length)		0000h	
Frame count	0100h		0100h	
Frame number	0100h		0100h	
Offset high	0000h		0000h	
Offset low	0000h		0000h	
Extended word 1	Slot Number	Slave Address	-	
Extended word 2	Alarm Spec Ack	Seq Number	-	
Extended word 3	Ext Diag	Alarm Type	-	
Extended word 4	-		-	

	Command	Response
Extended word 5	-	-
Extended word 6	-	-
Extended word 7	-	-
Extended word 8	Fault Information	-
Message Data byte 1	Data 1	
Message Data byte 2	Data 2	
Message Data byte 3	Data 3	
…	…	
Message Data byte n	Data n	

Slave Address

Station address of the slave the issued the alarm.

Slot Number

Used by the slave to indicate the source of the alarm. Range 0 to 254.

Seq Number

Unique identification number of the alarm. Range 0 to 31.

Alarm Spec Ack

Provides additional information about the alarm, such as an error appears or disappears. Also indicates whether the slave needs additional knowledge from the Master. For example, writing to a certain memory area with an Acyclic Write request.

Alarm Type

Identifies the alarm type such as Process Alarm, Plug Alarm, and so on. Range 1 to 6, 32 to 126.

Extended Diagnostic Flag

FFh: Slave sends an alarm message with "Extended Diag flag" set

00h: Slave sends an alarm message with "Extended Diag flag" cleared

Data [1 … n]

Additional manufacturer specific alarm information (Alarm - PDU)

Fault Information

If the Message Information word in the header of the message indicates "Invalid Other", addition information is available in this register.

3E00h: Module has received an invalid alarm indication data structure from a DPV1 slave ("Slave Address" contains the node address of the slave that issued the erroneous indication).

Refer to the PNO document "Extensions to EN50170 (DPV)" for more information on how to interpret these parameters.

6.2.9 Mailbox Message - Set Operating Mode

This command allows setting the operating mode of the module (that is, STOP, CLEAR, or OPERATE).

Parameter	Description
Command Initiator	Application
Command Name	SET OPERATING MODE
Command Number	0200h
Fragmented	No
Extended Header Data	Fault information may be returned in the header of the response.

Command and Response Layout - Set Operating Mode

	Command		Response		
Message ID	(ID)		(ID)		
Message information	0204h		0200h		
Command	0200h		0200h		Set Operation Mode
Data size	0000h		0000h		
Frame count	0100h		0100h		
Frame number	0100h		0100h		
Offset high	0000h		0000h		
Offset low	0000h		0000h		
Extended word 1	Conf Req	Req Mode	Conf. Req	Act. mode.	
Extended word 2	-		-		
Extended word 3	-		-		
Extended word 4	-		-		
Extended word 5	-		-		
Extended word 6	-		-		
Extended word 7	-		Appl. Specific Error Code		
Extended word 8	-		Fault Information		

Mode

40h: STOP

80h: CLEAR

C0h: OPERATE

Conf. Req.

00h: Confirmation is not required

01h: Confirmation required. All confirmation are automatically sent by the master, the user is not required to send a confirmation message.

Fault Information

If "Invalid Other" is returned in the Message Information word in the header of the response, information about the fault can be found here. Refer to ***Return Codes*** (page 134) for more information.

0100h: Invalid operating mode

FF00h: Module not initialized

6.3 Receiving Mailbox Message Responses from module

After a mailbox message has been sent, a response from the command, usually containing the requested data or the status of the command, is returned from the PTQ module to the processor. The response is returned from the PTQ-PDPMV1 via the PROFIBUS Input data block.

Note : This is for the original layout which the Default values, it changes if Slave diagnostics are chosen in PCB, User needs to refer to section 9.2.3.

Remembering the PROFIBUS Input Data Memory Map:

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
41101	%MW1101	0	Configuration, Status and Control data
-	-	-	
41163	%MW1163	73	Number of Messages in the In Mailbox Queue
41164	%MW1164	74	Number of Messages in the Out Mailbox Queue
41165	%MW1165	75	Number of Messages in the Alarm Queue
41157	%MW1157	76	Last Out Mailbox Message ID processed from Output Image
41158	%MW1158	77	Current In Mailbox Control Index
41159	%MW1159	78	Current Alarm Control Index
41180	%MW1180	79	Incoming Mailbox Message data
-	-	-	
41223	%MW1223	222	
41224	%MW1224	223	PROFIBUS Input Data
-	-	-	
41101+N+1	%MW1101+N+1	N	

The important section relevant to the Mailbox Messaging discussion is the Incoming Mailbox Data section (Word Offsets 79 to 222). Within this section of data, the following structure exists:

Mailbox Message Structure - From PTQ module

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Type	Description
41180	%MW1180	79	Message ID	Message ID value will match value used to generate the outgoing mailbox message
41181	%MW1181	80	Message Info	See individual commands for data values to be entered in each of these register locations
41182	%MW1182	81	Command	

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Type	Description
41183	%MW1183	82	Data Size	
44184	%MW1184	83	Frame Count	
41185	%MW1185	84	Frame Number	
41186	%MW1186	85	Offset high	
41187	%MW1187	86	Offset Low	
41188	%MW1188	87	Extended Word 1	
41189	%MW1189	88	Extended Word 2	
41190	%MW1190	89	Extended Word 3	
41191	%MW1191	90	Extended Word 4	
41192	%MW1192	91	Extended Word 5	
41193	%MW1193	92	Extended Word 6	
41194	%MW1194	93	Extended Word 7	
41195	%MW1195	94	Extended Word 8	
-	-	-	See individual commands	
41223	%MW1223	222		

Keep the following points in mind:

- If the In_Mailbox_Control_Index values are equal in the Input and Output Data blocks, the PTQ module will place the next message present in the mailbox queue into the Input Data image and increment the In_Mailbox_Control_Index in the Input Data image.
- After the processor processes a new In Mailbox Message, it should set the In_Mailbox_Control_Index (in the Output Image) to match the value received in the Input Image. This tells the PTQ module to transfer the next In Mailbox Message (if there is one) to the processor.

6.4 Mailbox Messaging Error Codes

6.4.1 Message Information

This register contains bit and code information about the mailbox message. The register is divided into five areas according to the following illustration:

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Message Type								ERR	C/R	(reserved)		Error Code			

Bit / Field	Description	Contents
ERR	This bit indicates if the received command contained any errors.	0: Message OK 1: Error (See also "Error Code" below)
C/R	This bit specifies whether the message is a command or a response.	0: Response Message 1: Command Message
Error Code	If the ERR bit is set, this field contains additional information about the error.	0h: Invalid Message ID 1h: Invalid Message Type 2h: Invalid Command 3h: Invalid Data Size 4h: Message header malformed (offset 008h) 5h: Message header malformed (offset 00Ah) 6h: Message header malformed (offset 00Ch to 00Dh) 8h: Invalid Response 9h: Flash Config Error Fh: Invalid Other (All other values are reserved)
Message Type	This field specifies the type of the message.	1h: Application Message 2h: PROFIBUS Specific Message 3h: Memory Message 5h: Reset Message (All other values are reserved)

6.4.2 Return Codes

Possible error codes in Message Data word "Return Code" *(The Return Codes can be byte swapped)*

Return Code	Name	Meaning
8010h	DPMC_ERR_V1C_CLOSED	Internal DPMC instance no longer exists.
8011h	DPMC_ERR_V1C_STOPPED	Internal DPMC instance has already been stopped
8012h	DPMC_ERR_V1C_STARTED	Internal DPMC instance has already been started

Return Code	Name	Meaning
8013h	DPMC_ERR_V1C_STATE_UNKNOWN	Internal DPMC instance has entered an undefined state
8021h	DPMC_ERR_V1C_REQ_ACTIVE	A request is already active
8022h	DPMC_ERR_V1C_NOT_ALLOWED	Internal DPMC module not initialized correctly
8023h	DPMC_ERR_V1C_INVALID_PAR	Invalid parameter in user request
8024h	DPMC_ERR_V1C_MEM_ALLOC	Internal memory allocation error
8025h	DPMC_ERR_V1C_L2_REQ	Unknown opcode in the confirmation
8026h	DPMC_ERR_V1C_TIMEOUT	Active request terminated with timeout
8028h	DPMC_ERR_V1C_INVALID_LEN	Invalid length in user request
8030h	DPMC_ERR_V1C_REQ_NEG[1]	Negative indication from lower layer
8031h	DPMC_ERR_V1C_REQ_RE	Message frame format error in response
8042h	DPMC_ERR_V1C_REQ_WITHDRAW	Request was recalled
8043h	DPMC_ERR_V1C_REQ_NOT_FOUND	Associated request block not found
80C1h	DPMC_ERR_V1C_MM_FE	Format error in request frame
80C2h	DPMC_ERR_V1C_MM_NI	Function not implemented
80C3h	DPMC_ERR_V1C_MM_AD	Access denied
80C4h	DPMC_ERR_V1C_MM_EA	Area too large
80C5h	DPMC_ERR_V1C_MM_LE	Data block length too large
80C6h	DPMC_ERR_V1C_MM_RE	Format error in response frame
80C7h	DPMC_ERR_V1C_MM_IP	Invalid parameter
80C8h	DPMC_ERR_V1C_MM_SC	Sequence conflict
80C9h	DPMC_ERR_V1C_MM_SE	Sequence error
80CAh	DPMC_ERR_V1C_MM_NE	Area non-existent
80CBh	DPMC_ERR_V1C_MM_DI	Data incomplete or incorrect
80CCh	DPMC_ERR_V1C_MM_NC	Master parameter set not compatible

Refer to ***Error Codes*** (page 135).

6.4.3 Error Codes

If return code indicates DPMC_ERR_V1C_REQ_NEG, the status values according to the DP-standard may be available in Error Code 1. Refer to the PROFIBUS DP specification for information on how to interpret these status values.

Error Code	Name	Meaning
01h	L2_STATUS_UE	
02h	L2_STATUS_RR	
03h	L2_STATUS_RS	
0Ch	L2_STATUS_RDL	Refer to PROFIBUS DP specification
0Dh	L2_STATUS_RDH	
0Fh	L2_STATUS_NA	

6.4.4 DPV1 Error Codes

Possible error codes in Message Data word "Return Code".

Return Code	Name	Meaning
0003h	DPMC_ERR_M_MEM_ALLOC	Internal memory allocation error
0004h	DPMC_ERR_M_L2_REQ	Unknown opcode in the configuration
0005h	DPMC_ERR_M_INVALID_PAR	Invalid parameter in user request
0007h	DPMC_ERR_M_NOT_IN_DATA	Slave is not in DataExchange (thus no DPV1 request can exist)
0012h	DPMC_ERR_M_REQ_ACTIVE	A request is already active
0018h	DPMC_ERR_M_NOT_ALLOWED	Internal DPMC module not initialized correctly
0021h	DPMC_ERR_M_CLOSED	Internal DPMC instance no longer exists
0022h	DPMC_ERR_M_STOPPED	Internal DPMC instance has already been stopped
0023h	DPMC_ERR_M_STARTED	Internal DPMC instance has already been started
0024h	DPMC_ERR_M_STATE_UNKNOWN	Internal DPMC instance has entered an undefined state
002Fh	DPMC_ERR_M_SLAVE_NOT_FOUND	Slave does not respond
0031h	DPMC_ERR_M_TIMEOUT	Active request terminated with timeout
0034h	DPMC_ERR_M_INVALID_LEN	Invalid length in user request
0035h	DPMC_ERR_M_REQ_NEG	Negative indication from lower layer
0036h	DPMC_ERR_M_REQ_RE	Message frame format error in response
0037h	DPMC_ERR_M_REQ_WITHDRAW	Request was recalled
0038h	DPMC_ERR_M_REQ_NOT_FOUND	Associated request block not found
0040h	DPMC_ERR_M_MM_FE	Format error in request frame
0041h	DPMC_ERR_M_MM_NI	Function not implemented
0042h	DPMC_ERR_M_MM_AD	Access Denied
0043h	DPMC_ERR_M_MM_EA	Area too large
0044h	DPMC_ERR_M_MM_LE	Data block length too large
0045h	DPMC_ERR_M_MM_RE	Format error in response frame
0046h	DPMC_ERR_M_MM_IP	Invalid parameter
0047h	DPMC_ERR_M_MM_SC	Sequence conflict
0048h	DPMC_ERR_M_MM_SE	Sequence error
0049h	DPMC_ERR_M_MM_NE	Area non-existent
004Ah	DPMC_ERR_M_MM_DI	Data incomplete or incorrect
004Bh	DPMC_ERR_M_MM_NC	Master parameter set not compatible
004Ch	DPMC_ERR_M_S7_XA	
004Dh	DPMC_ERR_M_S7_XR	PROFIBUS error for DPV1 (NRS-PDU received)
004Eh	DPMC_ERR_M_S7_XW	

7 Hot Standby Support

In This Chapter

- Hot Standby Overview 137
- Setting Up the Modicon Quantum Hot Standby with Unity System 142
- PTQ-PDPMV1 Operation 152
- PTQ-PDPMV1 Master Bus Properties when using a P&F DP/PA Segment Coupler 166

7.1 Hot Standby Overview

This section provides an overview of the PTQ-PDPMV1 PROFIBUS DP Master module specifications and startup support for Modicon Quantum Hot Standby system.

Use a Modicon Quantum Hot Standby with Unity system and PROFIBUS when downtime cannot be tolerated. Hot standby systems deliver high availability through redundancy. A hot standby system consists of two identical configurations.

- Modicon Quantum 140 CPU 671 60
- Modicon Quantum Power Supply Module
- Modicon Quantum RIO Head
- ProSoft Technology PTQ-PDPMV1 module hardware version 1.13 or higher
- Modicon Optional Modules (NOE, NOM)

One of the 140 CPU 67160s acts as the Primary controller and the other acts as the Standby controller. The Primary controller runs the application program and operates the remote I/O.

Note: The Modicon Quantum RIO Head is required even if the Remote I/O will not be used.

7.1.1 Identical Configurations

Two backplanes are configured with identical hardware and software. One of the programmable logic controllers (PLCs) functions as the Primary controller and the other as a Standby controller, and either controller can be put in the Primary state, but the other must be in the Standby state or offline.

7.1.2 Primary and Standby Controllers

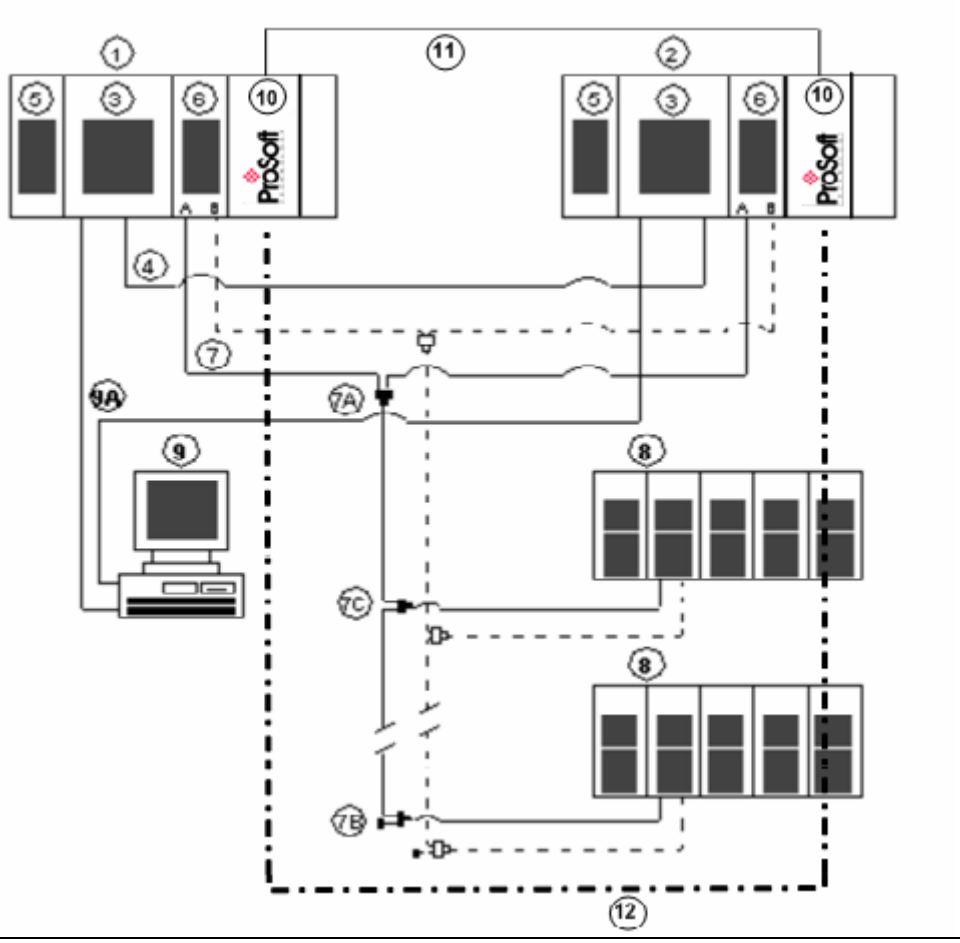
The Primary controller executes the application program, controls the remote I/O, and updates the Standby controller after every scan (program cycle). If the Primary controller fails, the Standby controller takes control within one scan. To determine if the Primary controller failed note controller's status displayed in the HE CPU LCD screen and the RIO Head's status displayed by the RIO Head's LEDs.

The Standby controller does not execute the full application program but only the first section, and the Standby controller does not control the remote I/O but checks out the availability of the Modicon Quantum Hot Standby with Unity equipment.

Note: For additional information on Quantum Hot Standby support refer to the UnityPro Hot Standby User Guide.

7.1.3 System Components

The following illustration shows the components required for a Modicon Quantum Hot Standby with Unity system.



1 Primary PLC
2 Standby PLC
3 Modicon Quantum Hot Standby with Unity controller with integrated coprocessor
4 Fiber Optic Cable to connect to both controllers
5 Modicon Quantum power supply module: Install power supply in first slot for better rack layout.
6 Modicon Quantum RIO head
7 Coaxial cable with splitters (7A) (MA-0186-100), trunk terminators (7B) (52-0422-000), and tap (7C) (MA-0185-100) for connecting the RIO heads (6) with the RIO drops (8). The dashed connections represent a redundant connection in the RIO network, which is not required for the Modicon Quantum Hot Standby with Unity system.
8 Modicon Quantum RIO drop
9 Unity Pro computer connected to both controllers via Modbus or Modbus Plus (9A)
10 PTQ-PDPMV1 HSBY modules
11 PTQ-PDPMV1 Ethernet redundancy communication cable
12 PROFIBUS network. Each PTQ-PDPMV1 placed at each network end.

Note: The 140 CRP 932 00 RIO Head unit will be required and present on the backplane for Hot Standby System to work.

7.1.4 Modicon Quantum Hot Standby with Unity and IEC Logic

Overview

A Modicon Quantum Hot Standby with Unity system requires two backplanes configured with identical hardware, software, and firmware. One of the controllers (PLC) functions as the Primary controller and the other as a Standby controller.

- The Primary updates the Standby after every scan.
- The Primary and Standby communicate constantly monitoring the health of the system.
- If the Primary fails, the Standby takes control within one scan.

7.1.5 Understanding System Scan Time in Modicon Quantum Hot Standby with Unity Systems

Effect on System Scan Time

The scan time of any Modicon Quantum Hot Standby with Unity system depends on the amount of data transferred. Because data must be transferred from Primary to Standby, any Modicon Quantum Hot Standby with Unity system always has a higher scan time than a comparable standalone system.

Performance Considerations

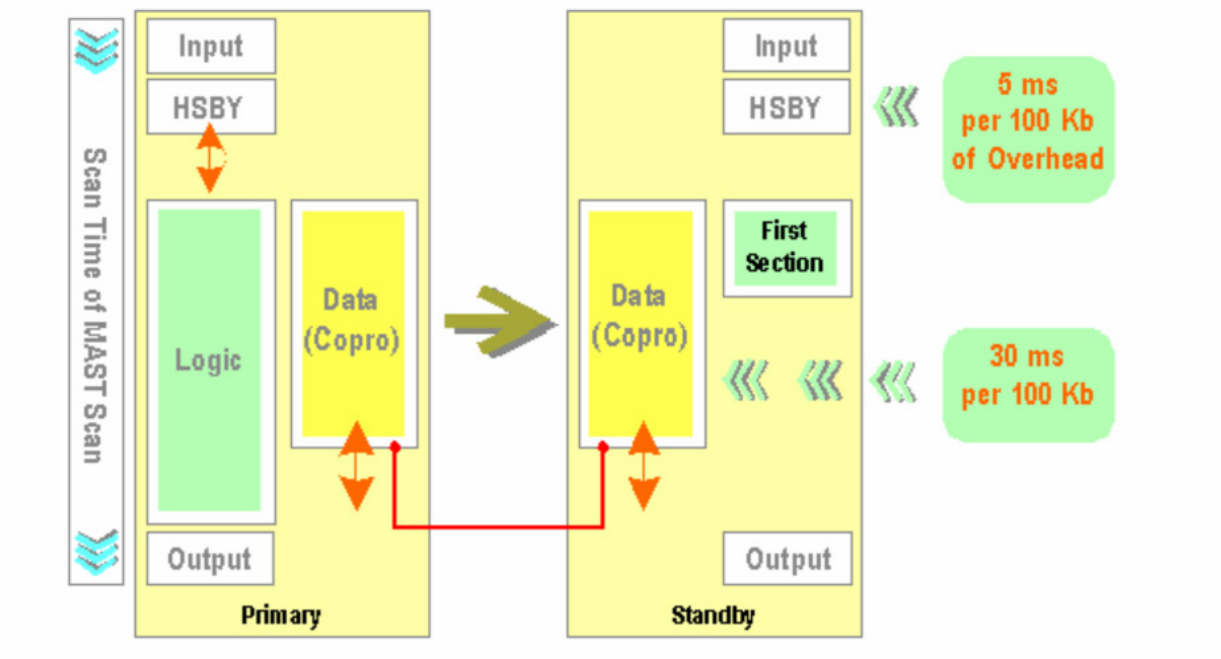
A Modicon Quantum Hot Standby with Unity system increases the length of a MAST scan, creating system overhead.

Note: System overhead is the time required to copy the application data to the communication link layer.

The network scan (communication between Primary and Standby "copros")

1 Exchanges data between both controllers
2 Runs in parallel with the application program.

A Hot Standby system



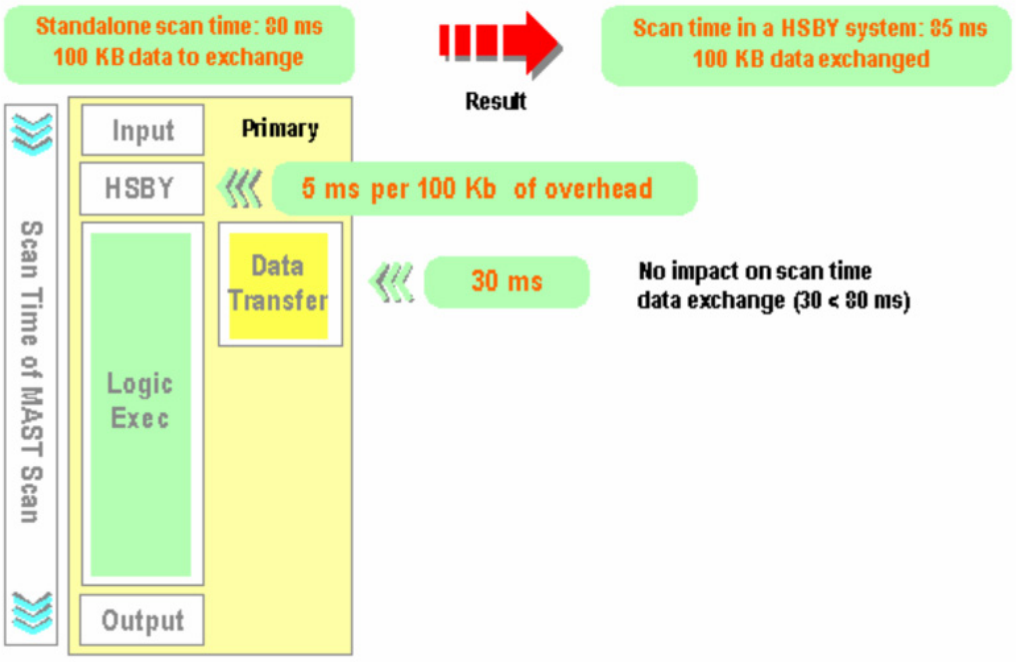
Most of time, the MAST scan hides the network scan.

Examples

However, when processing some application programs, additional system overhead may occur.

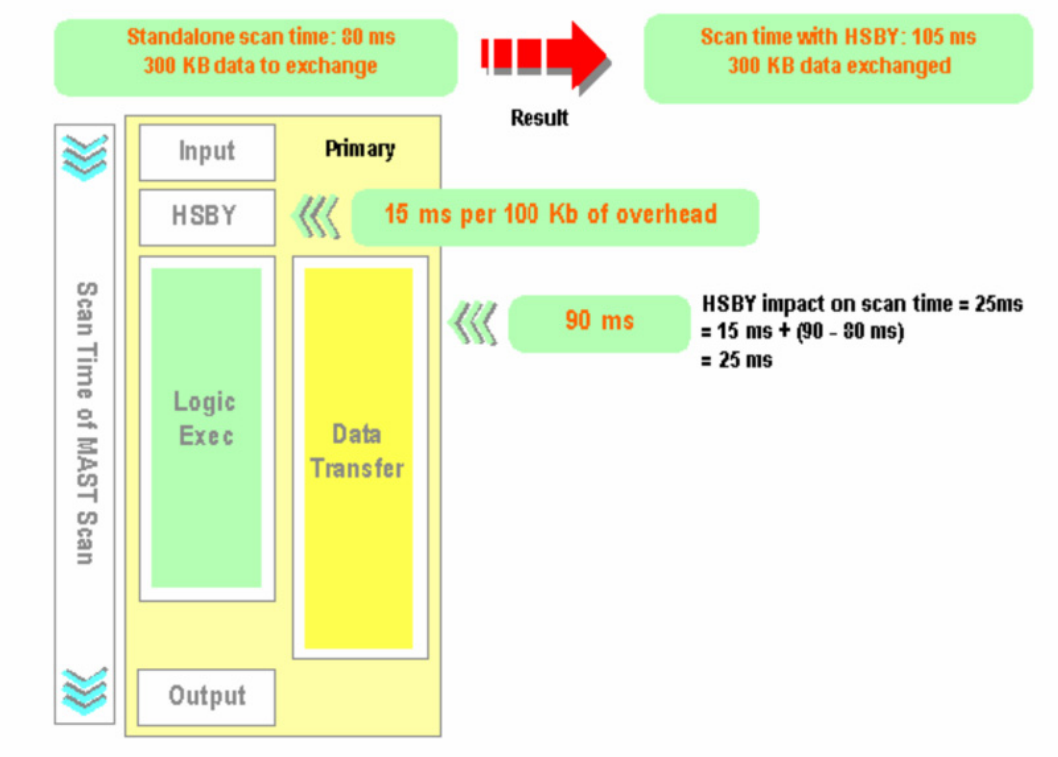
➢ *Example #1*

- Standalone application scan time: 80 ms
- Data (state RAM + unallocated variables): 100 Kb



➢ ***Example #2***

- Standalone application scan time: 80 ms
- Data (state RAM + unlocated variables): 300 Kb



Note: In addition to the above times for system overhead the PTQ-PDPMV1 module may acquire from 100 to 300ms of switch over time. All configured data is to be updated as fast as the combined asynchronous events occur based on the processor scan time, backplane transfer time, PTQ data handling time and PROFIBUS master bus cycle time. PROFIBUS Bus cycle time is based on slave reaction time, sync time, baud rate and other bus delays times for a given number of slaves on the network.

7.2 Setting Up the Modicon Quantum Hot Standby with Unity System

7.2.1 Overview

Schneider Electric is a leader in offering fault-tolerant, redundant systems, Hot Standby. Setting up a Modicon Quantum Hot Standby with Unity system involves a number of processes, summarized in the following paragraphs here and explained in detail elsewhere.

7.2.2 Mapping the Backplane Extensions

A Modicon Quantum Hot Standby with Unity requires two backplanes with at least four slots. You must map the two backplanes in an identical manner:

- same Modicon Quantum Hot Standby with Unity HE CPU with integrated coprocessor (Copro)
- same firmware
  - o same revision level
  - o same Modicon Quantum power supply module
  - o same Modicon Quantum RIO Head

And, if other modules are used, for example local I/Os, NOMs, NOEs, those modules must be identical.

For additional information on Modicon Quantum Hot Standby Startup support refer to the UnityPro User Guide.

7.2.3 PTQ-PDPMV1 Hot Standby Considerations

Limitations

The solution allows for one PTQ module per rack (Primary and Standby).

It will not be possible to install a PTQ in a RIO drop.

HSBY Operating Modes

Generally the user will have full control over the switchover via the command register through application program control. This is accomplished by the user application through the SW60 command register.

Glossary of Terms

- **Active master** – An active master is controlled by the Primary PLC and exchanges I/O data, diagnostics and parameter data with its assigned slaves.
- **Application –** If nothing else is stated, the term "application" refers to the application of the master module.
- **ASPC2** – Bus controller ASIC
- **Counter part –** The remote master.
- **CSI –** Crossed status information
- **FDL-layer** – Lower layer of a PROFIBUS communication stack (Layer 2 of the OSI model).
- **Hot Standby (HSBY**) – Refers to a redundant system with one Primary- and one Standby PLC where the Standby PLC is ready to take over if the Primary PLC fails.
- **PCB** (ProSoft Configuration Builder) – Software Configuration tool for the master module and PROFIBUS network.
- **Passive master** – A passive master is controlled by the Standby PLC and is ready to take over the communication with the slaves if the active master fails.
- **RTOS** – Real Time Operating System
- **SRD –** FDL-service for Send and Receive Data in one request.

- **Switchover** – A switchover occurs when the Standby PLC takes over control and becomes Primary.
- **TBD –** Short for "To be defined"

7.2.4 Hot Standby States

State Description

There are three normal running states of operation in a general Hot Standby system.

- **PRIMARY:** The PLC is set as the Primary CPU and is controlling Input/Output process as if it is stand-alone.
- **STANDBY:** The PLC is set as the Standby CPU and is ready to take over as Primary at all times, but the Primary CPU controls process and network. Outputs are not applied.
- **OFFLINE:** The PLC is set to Offline mode and the CPU cannot act like a Primary or Standby CPU.

It may be in STOP or disconnected mode. Here the PLC behaves as a normal non-HSBY CPU.

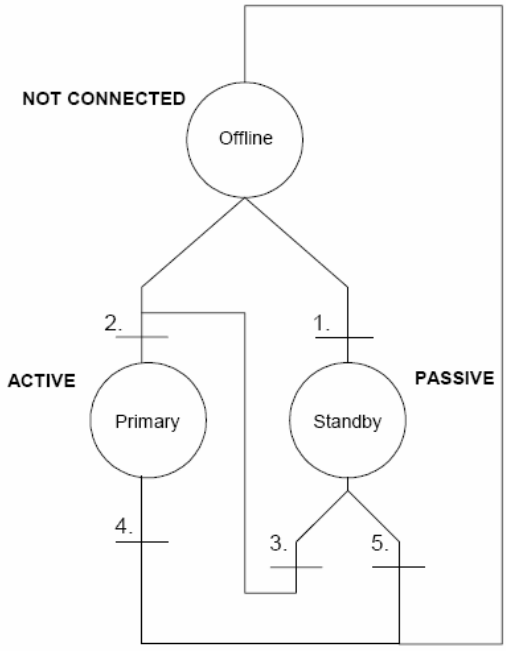


Figure 1 Hot standby states

The equivalent states for the master module are "Not connected", "Active" and "Passive" (bold text in the illustration above).

- **NOT CONNECTED**: The master module would be disconnected from the PROFIBUS network.

- **ACTIVE:** The master module would act as a class 1 PROFIBUS DPV1 master, managing I/O data, acyclic data, alarms, diagnostic- and parameter data with its assigned slaves as if it were stand-alone.
- **PASSIVE:** The master module would monitor the status of the active master, and if it detects any problems it would inform the application about the situation. Note that the passive master would not switch to active unless the application tells it to do so.

7.2.5 Transition Description

Offline to Standby (1)

- The passive master would attain its node address by subtracting one (1) from the address derived from the database. For example, if the master address in the database equals 2, the passive master would use address 1. If the master address in the database equals 0 it would use address 125.
- After a switchover it is important that the previously active master does not become passive before the counterpart has switched to active. If this statement is not adhered to, a dangerous situation with two masters having the same address (two passive masters in this case) would arise. The result of such a situation would be very unpredictable since there is really no way of detecting this situation.

Offline to Primary (2)

See Standby to Primary considerations.

Standby to Primary (3)

It is important that the master becomes active before the watchdog of the slaves expires. To allow for this switchover time the watchdog value would have to be extended in the bus parameter settings.

When the passive master switches to active it would change its node address to the primary address. To achieve this, the ASPC2 must be reset and reinitialized with the new bus parameter TS. Note that it is just the ASPC2 that is reinitialized, not the entire master module.

- The state of the "PA-bit" in register "HSBY Local status" would change to 0, indicating that the Master module now operates as active master.
- When the passive master switches to active it will not re-parameterize the slaves that report being in the "DATA" state.
- After a switchover the application will be informed when there is valid data available in the input output image area.
- Primary obtains the master node address.

Primary to Offline (4)

The active master leaves the bus as fast as possible since the risk of having two masters with the same address after a switchover must be avoided (two active masters in this case). The time it takes for leaving the bus will not exceed the switchover time.

Standby to Offline (5)

The time it takes to switch to offline is not critical since the transition would not influence the operation of the counter part, which will carry on working as a stand-alone master.

Note: For additional information and restrictions with Quantum processor behavior, refer to UnityPro HSBY User Guide.

7.2.6 HSBY State vs. Master Operation Mode

The matrix below indicates how the Master module would behave on a PROFIBUS network for all possible combinations of master operation mode (OFFLINE, STOP, CLEAR, OPERATE) and "HSBY state" (NOT CONNECTED, PASSIVE, ACTIVE).

	NOT CONNECTED	PASSIVE	ACTIVE
OFFLINE	No network traffic.	No network traffic.	
STOP		Ping requests are issued. Slave communication takes place	Ping requests are responded to. No slave communication takes place
CLEAR		The passive master cannot attain any of these operation modes. The application can however instruct the master to attain one of these modes after a switchover.	Ping requests are responded to. Slave communication takes place; only input data is read.
OPERATE			Ping requests are responded to. Slave communication takes place; both input- and output data is exchanged.

Table 1 HSBY state vs. Operation mode

7.2.7 Ping Message

The passive master would cyclically send ping messages to the active master, which in turn would respond to the message.

If the active master stops receiving ping requests (ping.req in figures below), it would assume that something is wrong with the counterpart or the field bus link.

If the passive master does not get any response (ping.res in figures below) to its ping requests it would assume that something is wrong with the counterpart or the field bus link. In both mentioned cases the erroneous situation would be signaled to the application by clearing (0) the COM-bit in the "HSBY Local Status" register.

The time between ping requests (TP) is bus cycle dependent. One request is sent every time the passive master is in possession of the token. If there is no response from the counterpart the ping message would be resent x times, where x is equal to bus parameter "max_retry_limit", before the COM-bit is cleared. The time the passive master waits for a response until it re-sends the message (TSL) is defined by bus parameter "SlotTime".

If the active master fails when it is in possession of the token, the passive master would sense this and reclaim the token after the timeout time TTO. In other words, the passive master will not wait for the entire TTR (Target Rotation time) to expire before it can send a ping request and detect the faulty active master.

TTO is calculated according to the following formula (according to the FDL-layer specification):

- TTO = 6*TSL + 2*TS*TSL, where TS is the physical address of the passive master.

The formula implies that the physical address of the passive master should be kept as low as possible in order to achieve an optimal time-out time.

To conclude, this means that the time it takes for the passive master to detect an active master failure (TFA) is based on two factors; the time it takes to reclaim the token + the time it takes to send a ping message with retries:

- TFA = TTO + ("max_retry_limit" + 1)*TSL

The active master would poll for incoming ping requests every TA ms, and if no request has been received since the last poll, the COM-bit is cleared. The poll sequence is asynchronous to the ping sequence, so in worst case it might take 2*TA (that is, the last poll took place just before the passive master failed) before the active master detects a failure.

TA is calculated according to the following formula:

- TA = Max[30, TTR+SM], where SM is a safety margin (10% of TTR).

The formula implies that the minimum time between polls is 30 ms (highest timer resolution of the RTOS is 5 ms.), while the maximum time is proportional to the Target Rotation time (TTR).

The time it takes for the active master to detect a passive master failure is not so crucial since no switchover would take place. The active master would just carry on operating as a stand-alone master. Another argument to keep TA at a reasonable value is that we would like to keep the CPU-load as low as possible since the most important thing for the active master is to keep up the pace with its assigned slaves.

The table below gives an example of the discussed timing values for some baud rates.

Preconditions

The values are based on a PCB PROFIBUS Master Configuration Software configuration consisting of 96 slaves, where each slave has 16 bytes input data and 16 bytes output data (that is, max DPRAM size). The bus profile in the bus parameters set-up is set to "Single master". The active master has a physical address equal to one, which means that the passive one will use address zero. TTR in the example is doubled compared to the value that the master configuration software calculates.

Baud Rate	TA[ms]	TP[ms]	TFA[ms]
12 Mbit/s →T_{TR} = 50 ms, T_{SL} = 1000 Tbits	55	≤ T_{TR}	1
1.5 Mbit/s →T_{TR} = 186 ms, T_{SL} = 300 Tbits	205		2
500 kbit/s → T_{TR} = 500 ms, T_{SL} = 200 Tbits	550		4
93.75 kbit/s (PA) → T_{TR} = 2374 ms, T_{SL} = 100 Tbits	2611		9
45.45 kbit/s (PA) → T_{TR} = 14000 ms, T_{SL} = 640 Tbits	15400		113
9.6 kbit/s → T_{TR} = 23200 ms, T_{SL} = 100 Tbits	25520		83

Table 2 Calculated times for detecting a missing counterpart

Ping Sequence

This section shows the ping sequence, and how the COM-bit in register "HSBY Remote status" is affected under different scenarios.

The figures show the active- and the passive master connected via the PROFIBUS network. For each master the DPRAM (that is, application interface) and the software objects that are responsible for the ping sequence is shown.

Start-up

Active master operational prior to passive

As soon as the active master is ready to communicate on the PROFIBUS network it would start polling for ping messages from the passive one. As soon as it receives the first ping request the COM-bit is altered from zero to one.

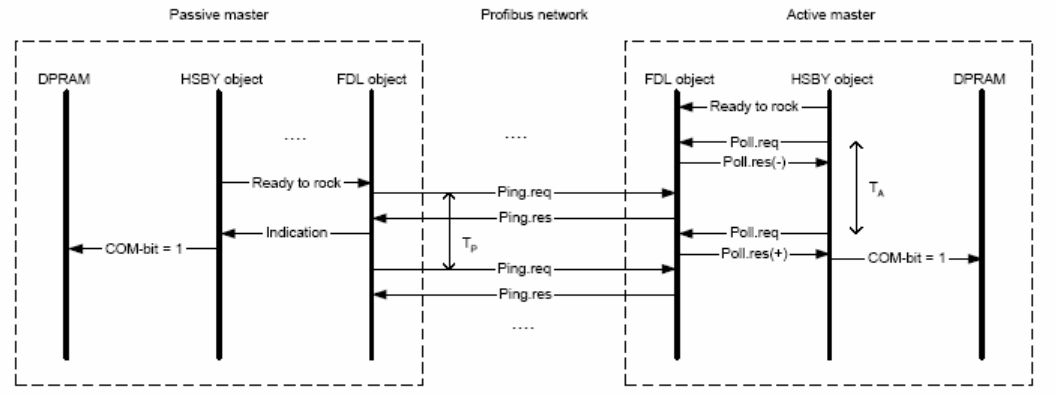


Figure 2 Active master is ready prior to Passive master

Passive master operational prior to active

As soon as the passive master is ready to communicate on the PROFIBUS network it would start sending ping messages to the active one. As soon as the active master responds the COM-bit is altered from zero to one.

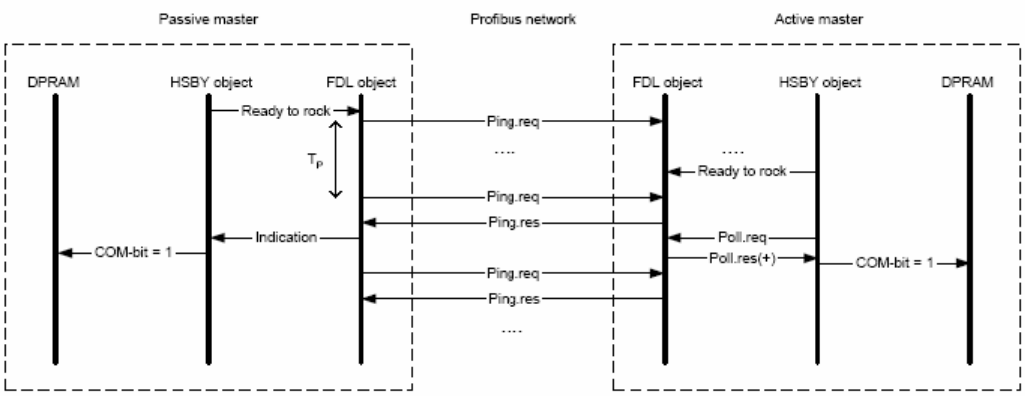


Figure 3 Passive master is ready prior to Active master

Passive Master Failure

When the active master has not received any ping messages from the counterpart within TA it would clear the COM-bit. This scenario would also apply when the field bus link between the two masters is lost due to a cable break for example.

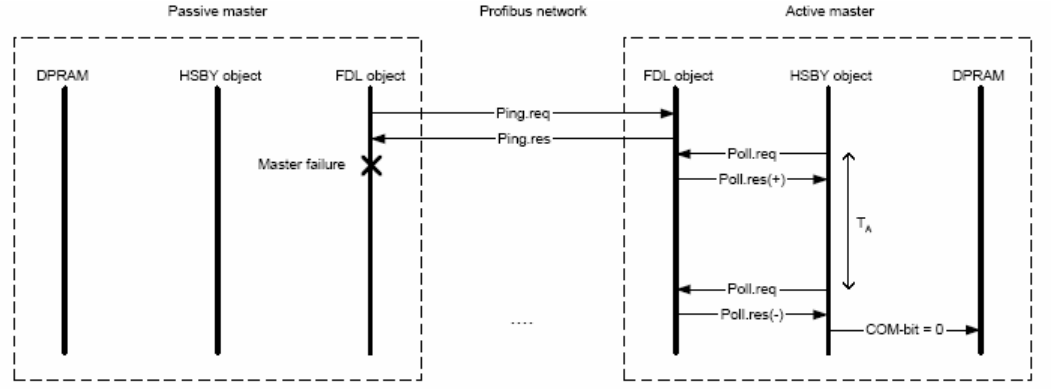


Figure 6 Passive master failure

Ping Message Structure

In addition to detecting a lost counterpart, the ping message is also used to communicate status information between the active and passive master.

The ping message would carry the following data (4 bytes).

Byte	Data	Description
0	HSBY Local status	Status information of the passive master.
1	HSBY Nr of local slaves	Number of slaves accessible to the passive master.
2 to 3	16-bit CRC	CRC of the passive masters database.

Table 3 Ping.req

Byte	Data	Description
0	HSBY Local status	Status information of the active master.
1	HSBY Nr of local slaves	Number of slaves accessible to the active master
2 to 3	16-bit CRC	CRC of the active masters database.

Table 4 Ping.res

7.2.8 PTQ Link Message

The PTQ modules require Ethernet UDP services for active and passive module communications.

Two types of services are provided.

1 Service port 3001 – Used for status data and CRC data of the standby to primary unit. (The CRC values are listed below from the PTQ-PDPMV1 Reference Guide)

Quantum Address	Unity Address	Word Offset	Name	Description
1043	%IW1043	43	PROFIBUS configuration Checksum	CRC32 checksum for PROFIBUS Master configuration downloaded from configuration utility
1045	%IW1045	45	PTQ Module configuration checksum	PTQ-PDPMV1 module configuration checksum for module configuration downloaded from configuration utility

1 Service port 3002 – Used for DPV1 messages. The transfer of these messages is necessary to insure the messages are delivered and received upon a switchover condition.

7.2.9 Crossed Status Information

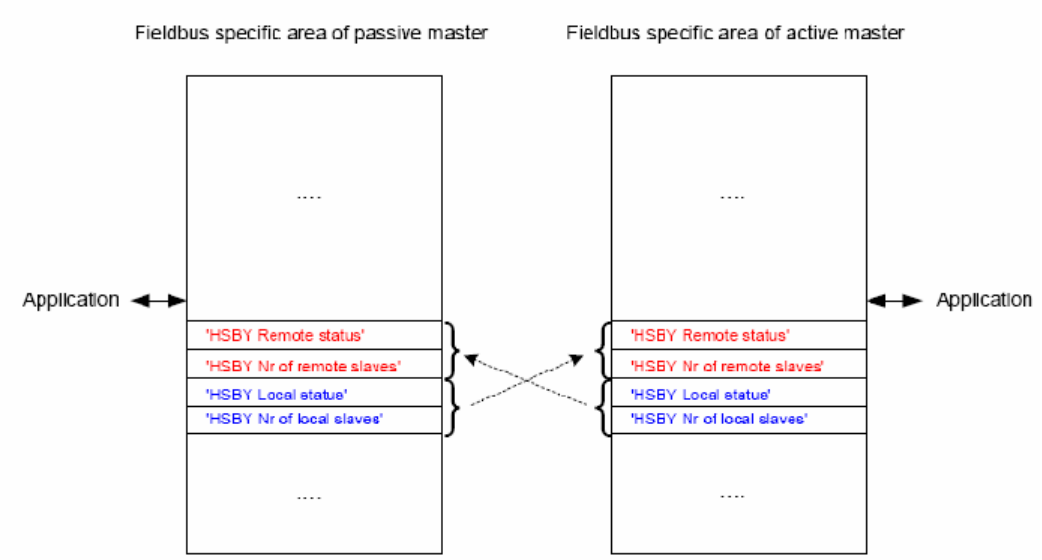
The ping message communicates status information between the active- and passive master, and vice versa.

A part of this "crossed status information" (CSI) would be presented in the "HSBY Remote status"-/"HSBY Nr of remote slaves" registers located in the fieldbus specific area of the DPRAM.

These registers would be updated every time new status information is received from the counterpart. Polling for new status information takes place every TA ms.

Normally, the application would forward this information to a high-level system (for example, application of the primary PLC), which in turn would determine if a switchover should be carried out, or not.

The following illustration shows how CSI is communicated between the two masters and how it is displayed in the fieldbus specific area of the DPRAM.



7.2.10 Conditions for Switchover

Each PTQ situated in the primary or the standby local rack must provide both CPUs with its own diagnostics in order to request and perform a switchover. Diagnostics must be crossed between Primary and Standby PTQs and the associated CPU. Both CPUs (Primary and Standby) must be informed anytime of the status of all PTQs. The PTQ module will post Local and Remote status information in the Input Status/Control Data Area block word offset 60 to 63.

Note: for the backplane driver, the 100ms to 300ms spec depends upon scan time. The backplane driver cannot detect a switch over in 300ms if the scan time is 500ms. The backplane drivers can communicate with the PLC only at the end-of-scan.

In other words:

- The Primary must be informed of the status of its own PTQ(s),
- The Primary must inform the Standby of the status of its own PTQ(s),
- The Standby must be informed of the status of its own PTQ(s),
- The Standby must inform the Primary of the status of its own PTQ(s).

Conditions for user to consider switch over are as follows:

- PTQ master Module failure
- Bus not connected or all devices not responding
- PTQ master module not configured
- Bus cable break. Status information provided to user to determine appropriate master with most slaves will be available in the Input data block.

Based on this crossed information updated anytime and simultaneously, the user application code can perform a switchover using command register and status bits. A switchover can be performed only if the Primary PTQ fails and the Standby PTQ is able to take control. We cannot allow the system to switchover if the Standby PTQ status is not known with precision. In case of several PTQs per rack, the application code will have to diagnose the health status of all Standby PTQs before performing a switchover. These diagnostics have to be taken into consideration in order to avoid leaving a bad situation on one side (Primary) and getting a worse one on the other side.

7.3 PTQ-PDPMV1 Operation

While in primary mode the module will read the output area and write the input area. The module will constantly scan the HSBY control word to determine the HSBY state.

- Active LED will flash if the module is in Standby mode in a Hot Standby system. The Hot Standby Status Word of the Modicon Quantum processor is read during each end-of-scan.
- During switchover caused by a failure on the primary, all PROFIBUS I/Os will be held at their last values, till the moment the new PTQ takes control (No glitch on I/O devices).

7.3.1 PTQ-PDPMV1 HSBY Diagnostic Data

PTQ Input Data Block

The PTQ module reports the HSBY Local and Remote Status Registers via the Input Data Block.

Input Data Block *(HSBY words only)*

Quantum Address (Example)	Unity Address (Example)	Word Offset	Name	Description
1060	%IW1060	60	Low byte: HSBY Passive Status - from PROFIBUS interface High byte: HSBY Passive number of slaves - from PROFIBUS interface	

High Byte								Low Byte							
16				8				7				0			
								COM	--	--	OD	DB	CE	SO	PA

Bit	Explanation
PA	This bit would indicate the state of the local master. 0 – Active master. that is, master is controlled by the Primary PLC. 1 – Passive master. that is, master controlled by the Stand-by PLC.

<table>
<tr><th>Quantum Address (Example)</th><th>Unity Address (Example)</th><th>Word Offset</th><th>Name</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td></td><td>
<table>
<tr><td>SO</td><td>This bit indicates if the local master recognizes any of its assigned slaves as "offline".
0 – At least one slave is "offline"
1 – All slaves OK</td></tr>
<tr><td>CE</td><td>This bit indicates if the local master has recognized a critical error.
0 – No critical errors
1 – Critical error active
This bit is set when problems with the ping sequence is encountered.</td></tr>
<tr><td>DB</td><td>This bit indicates if the local master has detected a database mismatch.
0 – Database OK
1 – Database mismatch</td></tr>
<tr><td>OD</td><td>This bit indicates when the data in the Output data area of the DPRAM is updated after a switch over.
0 – Output data is not updated
1 – Output data is updated (Once this bit is set, it remains set for the remaining session until the Anybus is either reset or HSBY state changes to "Not Connected")</td></tr>
<tr><td>-</td><td>Not used set to zero</td></tr>
<tr><td>-</td><td>Not used set to zero</td></tr>
<tr><td>COM</td><td>This bit indicates if the counterpart is present.
0 = Counterpart not present
1 = Counterpart is present</td></tr>
</table>
</td></tr>
<tr><td>1061</td><td>%IW1061</td><td>61</td><td>Low byte: HSBY Active Status - from PROFIBUS interface
High byte: HSBY Active number of slaves - from PROFIBUS interface</td><td>
<table>
<tr><th colspan="8">High Byte</th><th colspan="8">Low Byte</th></tr>
<tr><td colspan="8">16</td><td colspan="8">8</td><td colspan="8">7</td><td colspan="8">0</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>HS</td><td>-</td><td>-</td><td>OD</td><td>DB</td><td>CE</td><td>SO</td><td>PA</td></tr>
</table>
<table>
<tr><td>HS</td><td>This bit indicates that the Hot standby functionality is enabled.
0 – HSBY disabled. Module operates as "stand alone" master or HSBY-state equals "Not connected".
1 – HSBY enabled</td></tr>
</table>
See explanation of bits OD, DB, CE, SO and PA in chart for address 1060.
</td></tr>
<tr><td>1062</td><td>%IW1062</td><td>62</td><td>HSBY Message length - from UDP HSBY Server</td><td></td></tr>
</table>

Quantum Address (Example)	Unity Address (Example)	Word Offset	Name	Description
1063	%IW1063	63	Low byte: HSBY Passive Status - from UDP HSBY Server High byte: HSBY Passive number of slaves Message length - from UDP HSBY Server	Refer to word 60 for explanation. This is a backup word derived from Ethernet UDP messaging
1064 to 1165	%IW1064 to %IW1165	64 to 65	HSBY Passive PROFIBUS CRC32 - from UDP HSBY Server	CRC32 checksum for PROFIBUS Master configuration downloaded from configuration utility via UDP
1066 to 1167	%IW1066 to %IW1167	66 to 67	HSBY Passive User Cfg CRC32 - from UDP HSBY Server	PTQ-PDPMV1 module configuration checksum for module configuration downloaded from configuration utility via UDP

New HSBY Input Status Data Word Details

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
1060	%IW1060	60	Active HSBY Master Status data and number of slaves seen by this master
1061	%IW1061	61	Passive HSBY Master Status data and number of slaves seen by this master

Word offset 60 HSBY Active Master Status data

HSBY Active # of slaves (High byte)		HSBY Active Status (Low byte)	
15	8	7	0

Word offset 61 HSBY Passive Master Status data

HSBY Passive # of slaves (High byte)		HSBY Passive Status (Low byte)	
15	8	7	0

Active and Passive HSBY Master Status data Low byte – bits 0 to 7

Note. Bits 0 to 7 are not considered valid until the "HS-bit" equals one, that is, Master module is initialized as a HSBY master (Passive or Active). Once set, the "HS-bit" will keep this value for the remaining active session until the Master module is either reset or HSBY-state changes to "Not connected".

All bits would be set to zero at power-up and when the HSBY-state equals "Not connected".

Bit	Name	Explanation
0	PA	This bit indicates the state of the local master. 0 = Active master: master is controlled by the Primary PLC 1 = Passive master: master is controlled by the Standby PLC
1	SO	This bit indicates if the local master recognizes any of its assigned slaves. 0 = At least one slave is "offline" 1 = All slaves OK

Bit	Name	Explanation
2	CE	This bit indicates if the local master has recognized a critical error. 0 = No critical errors 1 = Critical error active This bit is set when problems with the ping sequence are encountered. Detailed information about the problem is dumped in the fieldbus specific area at address hF90-FBC.
3	DB	This bit indicates if the local master has detected a database mismatch. 0 = Database OK 1 = Database mismatch.
4	OD	This bit indicates when the data in the Output area of the DPRAM is updated after a switch over. 0 = Output data is not updated 1 = Output data is updated. (When this bit is set, it remains set for the remaining session until the Anybus is either reset or HSBY state changes to "Not Connected")
5	-	Not used, set to zero.
6	-	Not used, set to zero.
7	HS	This bit indicates that the Hot Standby functionality is enabled. 0 = HSBY disabled. Module operates as "stand-alone" master or HSBY-state equals "Not Connected". 1 = HSBY enabled.

Table 5 HSBY Local status

Active and Passive HSBY Master Status data High byte – bits 8 to 15

The "COM-bit" would be set to one when the counterpart is present on the network (that is, ping sequence is successfully running). The other bits (8 to 14) are considered valid only when the "COM-bit" equals one.

If the ping sequence is terminated the "COM-bit" and all other bits (8 to 14) are set to zero.

All bits would be set to zero at power-up and when the HSBY-state equals "Not connected".

HSBY Nr of Active slaves byte

This byte will indicate the number of slaves accessible to the local master. Based on this information the high-level system could switch to the master that recognize most slaves.

Note that this register only contains valid data when the "HS-bit" in register "HSBY Local status" is set.

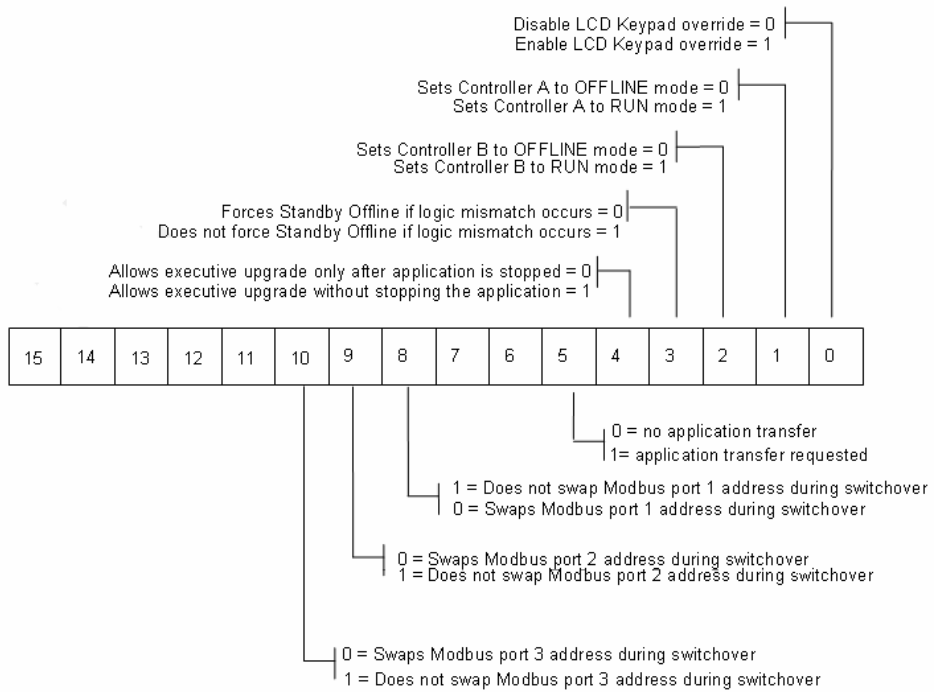
HSBY Nr of Passive slaves byte

This byte will indicate the number of slaves accessible to the counter part. (that is, a reflection of the counterparts "HSBY Nr of local slaves" register). Based on this information the high-level system could switch to the master that recognize most slaves.

Note that this register only contains valid data when the "COM-bit" in register "HSBY Remote status" indicates that the counter part is present. When the "COM-bit" is cleared this register would be set to zero.

PLC HSBY Command Register

The following illustration identifies the operating options provided by the Command Register. The Command Register is used to define the operating parameters of a Hot Standby application. The Command Register is %SW60. The Command register defines the operation of the Hot Standby application. That means both the Primary and Standby. Therefore the Command Register is transferred to the Standby PLC each scan. As a result, any changes made to the Command Register on the Standby PLC will have no effect since the value transfer from the Primary side will overwrite it.


Disable LCD Keypad override = 0
Enable LCD Keypad override = 1
Sets Controller A to OFFLINE mode = 0
Sets Controller A to RUN mode = 1
Sets Controller B to OFFLINE mode = 0
Sets Controller B to RUN mode = 1
Forces Standby Offline if logic mismatch occurs = 0
Does not force Standby Offline if logic mismatch occurs = 1
Allows executive upgrade only after application is stopped = 0
Allows executive upgrade without stopping the application = 1
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 = no application transfer
1= application transfer requested
1 = Does not swap Modbus port 1 address during switchover
0 = Swaps Modbus port 1 address during switchover
0 = Swaps Modbus port 2 address during switchover
1 = Does not swap Modbus port 2 address during switchover
0 = Swaps Modbus port 3 address during switchover
1 = Does not swap Modbus port 3 address during switchover

%SW60 Hot Standby Command Register

%SW60.0: This bit, if set to 1, allows the Command Register RUN status of the PLC to be set through the LCD Keypad.

Warning; If the Keypad override is enabled while the Hot Standby system is running, the Primary PLC will immediately read bits 14 and 15 to determine its own state and the state of the Standby. If both bits are set to 0, a switchover will occur and the former Primary will go offline. The new Primary will continue to operate.

%SW60.1: Setting this bit = 1 will put PLC A in the run mode. Setting the bit = 0 will put PLC A in Offline mode. This bit has effect only if bit 16 is set = 1.

%SW60.2: Setting this bit = 1 will put PLC B in the run mode. Setting the bit = 0 will put PLC B in Offline mode. This bit has effect only if bit 16 is set = 1.

%SW60.3: Setting this bit = 0 will force the Standby PLC Offline if a logic mismatch is detected. Logic mismatch is defined as either the MID, LID or CID being different on Primary and Standby sides. Setting this bit = 1 will allow the Standby PLC to continue to operate normally even if the MID is different on the Primary and Standby.

%SW60.4: Setting this bit = 1 allows the executive to upgrade on the Standby without having to stop the application. This means the Hot Standby system is allowed to operate with different versions of the OS running on the Primary and Standby. This option is provided to allow upgrades to be done without shutting down the process. Clearly the Standby PLC must be stopped to do the executive upgrade, but it will be able to operate as a valid Standby when started again.

%SW60.5: Setting this bit = 1 commands the standby station to initiate an application transfer. That function is not required in UNITY V1.

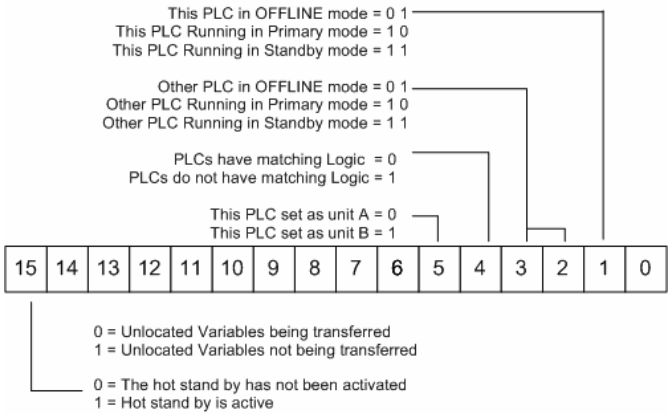
%SW60.8: If this bit set = 1 the Modbus address on port 1will be swapped when a switchover occurs. Swapped means to change address by □ 128 to keep the address in the range of 1 to 247. The purpose of this is to allow the P-unit of an HMI to always use the same address to connect to the Primary or Standby/Offline PLC.

PLC HSBY Status Register

The Status Register provides user information relative to the state of the two PLCs in the Hot Standby system. The Status Register is %SW61.

Both the Primary and Standby/Offline PLCs have their own copy of the Status register. The Status register is not transferred from Primary to Standby each scan. Each PLC must maintain its local Status Register based on the regular communication between the two PLCs.

The following illustration identifies the operating options provided by the Command Register.


This PLC in OFFLINE mode = 0 1
This PLC Running in Primary mode = 1 0
This PLC Running in Standby mode = 1 1
Other PLC in OFFLINE mode = 0 1
Other PLC Running in Primary mode = 1 0
Other PLC Running in Standby mode = 1 1
PLCs have matching Logic = 0
PLCs do not have matching Logic = 1
This PLC set as unit A = 0
This PLC set as unit B = 1
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 = Unlocated Variables being transferred
1 = Unlocated Variables not being transferred
0 = The hot stand by has not been activated
1 = Hot stand by is active

%SW61 Hot Standby Status Register

The following gives additional detail on the various part of the Status Register.

%SW61.0 to 3: These bits display the state of the local and remote Hot Standby PLC.

%SW61.4: This bit is set = 1 whenever a logic mismatch is detected between the Primary and Standby PLCs. This means that either the MID, CID or LID is different on the two PLCs. Under this condition, if bit 13 of the Command Register is set = 0, bit 1 of the Status Register will be set = 1.

%SW61.5: This bit identifies the order reported by the copro at start time depends on the range of the MAC addresses.

- If the A/B designation is A, then bit 5 will be set = 0.
- If the A/B designation is B, then bit 5 will be set = 1.

%SW61.14: If set = 1 it indicates that a logic mismatch has been detected that disallows Unlocated Variables to be transferred from Primary to Standby. This feature had been cancelled for UNITY V1.1. A switch over with a partial application context had been classified as too much dangerous against people and goods.

%SW61.15: If set = 1 it indicates that the Copro device is set up correctly and working.

How the PTQ detects when a switch over is detected

PTQ and master bus scanner must change state of behavior when the PLC changes its state within 100 ms not to exceed 300 ms. PTQ reads every scan the PLC status in word 102 (%SW61) of the configuration table.

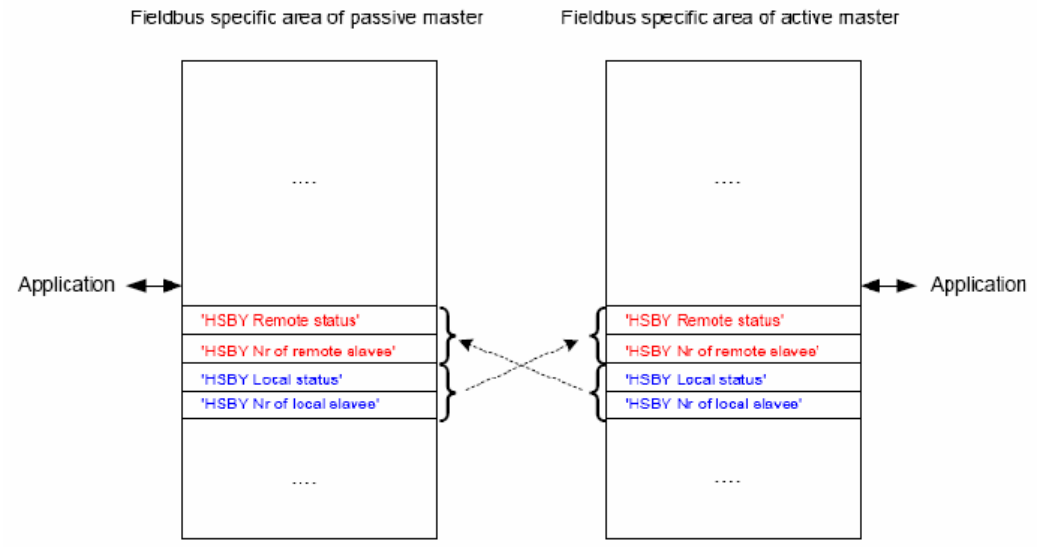
Crossed status information

The ping message communicates status information between the active- and passive master, and vice versa.

A part of this "crossed status information" (CSI) would be presented in the "HSBY Remote status HSBY Nr of remote slaves" registers. These registers would be updated every time new status information is received from the counterpart. Polling for new status information takes place every TA ms.

The PTQ module application will forward this information to Quantum processor which in turn would determine by user application if a switchover should be carried out or not.

The following illustration shows how CSI is communicated between the two masters and how it is displayed in the Field bus specific area of the DPRAM.



Slave status

Both masters inform the counterpart of its slave status as a part of the CSI.

The Slave status information includes two parts, the SO-bit in register "HSBY Local status", and register "HSBY Nr of local slaves". Based on this information a high-level system could switch to the master that recognize most slaves.

- The "SO-bit" indicates if the master recognizes any of its assigned slaves as "offline".
- Register "HSBY Nr of local slaves" indicates how many slaves are "online".

How this information is determined depends on if the Master module operates as active- or passive master:

- The active master would use the "state-report" information available from the Siemens stack. A slave is considered to be "online" when it participates in the cyclic "DataExchange" sequence.

Note. This means that a slave that reports "Prm-fault" or "Config-fault" is considered to be "offline" even if it is physically accessible to the master.

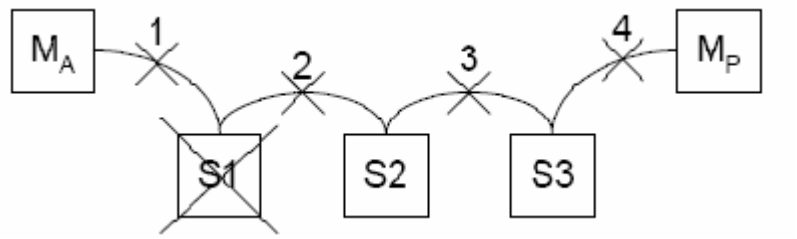
- The passive master would ping all of its assigned slaves with FDL_Status telegrams once every time it holds the token. A slave is considered to be "online" when it responds to the telegram.

During a switchover the Slave status information would be reassembled. This means that the value of "HSBY Nr of local slaves" and "SO-bit" might dip for a short while until the "new" active master has detected its slaves.

Note: One must be aware of that this feature affects the performance of the entire network, since the passive master must query all slaves when it holds the token, thus increasing the token rotation time.

Example:

The following illustration shows a network with three slaves and two masters. MA is the active master and MP the passive one.



The table lists the values of the slave status information under different scenarios.

Note the only time the slave status information is forwarded to the counterpart is when a slave is disconnected or fails. In all other cases the communication link is broken so the information would not reach the counterpart.

	MA		MP	
Type of Failure	Local "SO-bit"	"HSBY Nr of local slaves"	Local "SO-bit"	HSBY Nr of local slaves
No errors	1	3	1	3
Cable is cut at 1.	0	0	1	3
Cable is cut at 2.	0	1	0	2
Cable is cut at 3.	0	2	0	1
Cable is cut at 4	1	3	0	0
S1 is disconnected	0	2	0	2

Table 7 Slave status example

Database Mismatch

Both masters would inform the counterpart of its database CRC value as a part of the Crossed Status Information.

Based upon this information both masters would compare its own CRC value with the one received from the counterpart to determine the state of the "DB-bit" in register "HSBY Local status".

It does not make sense to perform the CRC check in a cyclic manner. Instead, it would be enough to do it one time when the counterpart has just been detected.

FDL Layer Access

Ping messages, Slave status messages and DPV1 status messages would be communicated over the PROFIBUS network via the FDL-layer of the Siemens stack (a.k.a. AMPRO2).

A unique channel, reserved only for Ping- and Slave status messages, would be opened to the FDL-layer.

This would ensure that no "DPV1 class 2"- or "Live List" requests, which also use the FDL-layer, are interfering with the time critical ping-/slave status sequence.

Message type and priority

- Ping messages would be sent as high-priority SRD-repeat telegrams.
- Slave status messages would be sent as low-priority FDL_Status-repeat telegrams.
- DPV1 status messages would be sent as low-priority SRD (single) telegrams.

SAP-number

- SAP 10 would be used for Ping messages.
- SAP 11 would be used for DPV1 status messages.
- No SAP is defined for Slave status messages (FDL_Status).

The following table lists FDL-services that would be needed in order to accomplish the HSBY functionality. The two columns to the right indicate which services that are used by the active- and passive master.

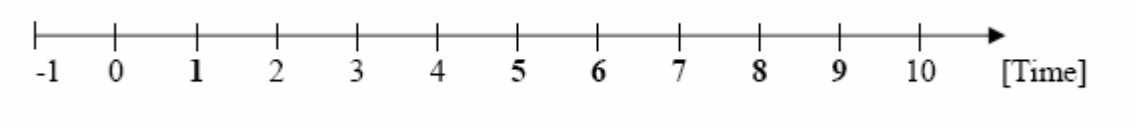
FDL-service	Description	Active master	Passive master
Open-Channel	Establish a channel to the FDL-layer	X	X
SAP-Activate	Opens up a Service Access Point at the responder	X	X
IND-Resource-Provide	Makes a resource available for single SRD-requests		X
IND-Resource-Repeat-Provide	Makes a resource available for SRD-repeat-requests	X	
Reply-Update-Multiple	Updates the SRD-response data	X	
FLC-Repeat-Exchange	Reads out the latest SRD-request data	X	
SRD	Sends an SRD-request (used for DPV1 status messages)	X	
SRD-Repeat	Sends an SRD-request cyclically (used for Ping messages)		X
MAC-Repeat-Exchange	Exchanges SRD-request or SRD-response data		X
MAC-Reset	Resets ASPC2 during switchover		X
FDL_Status-repeat	Sends an FDL_Status request cyclically (Used for Slave status messages		X

SAP-number 10 would be used for the DSAP of the responder (active master) as well as for the SSAP of the initiator (passive master).

The SRD-request would be sent as a "low priority" FDL-message.

7.3.2 Time for Switchover

The overall switchover time for a Hot Standby system is dependent on several sequential events, which are outlined in the timeline below. The elapsed time between some events is not static but highly dependent on the baud rate, these events are written in **bold text**.



–1	Last DataExchange request before active master failure (that is, last re-triggering of "slave watchdog")
0	Active master failure.
1	Passive master detects counterpart failure.
2	Passive master indicates the situation by clearing the COM-bit in Fieldbus specific area.
3	Application of the passive master detects that the active master has failed and forwards this information to a higher-level system for further processing.
4	Application of the passive master initiates a switchover based on the decision from a higher-level system.
5	Passive master performs the switchover (MAC reset with new TS)
6	Active master (former passive) issues an initial GlobalCtrl request (that is, first PROFIBUS telegram after reinitialization of MAC).
7	Active master has issued a SlaveDiag request to all slaves (first re-triggering of "slave watchdog" after switchover).
8	Active master has issued a DataExchange request to all slaves (second re-triggering of "slave watchdog").
9	Active master sets the OD-bit in the Fieldbus specific area indicating that there is fresh data in the output data area for the application to read.
10	Application of the active master detects that the OD-bit is set and accesses the output area.

7.3.3 Bus Parameters

Some bus parameters need to be altered to allow for Hot Standby functionality.

Slave watchdog time (WD_Factor1&2)

The watchdog time of the slaves would have to be increased to allow for the maximum time it takes for the HSBY system to perform a switchover.

Token rotation time (TTR)

To allow for master-master communication TTR would have to be increased by a factor 2 compared to a single master configuration.

Highest Station Address (HSA)

HSA defines the highest address that a master can have to be included in the token ring. For example, a master with address 35 will not be included in the token ring if HSA is set to 34.

It is recommended to keep HSA as low as possible since it affects the time it takes for a master to enter the token ring.

Master address (TS)

The address of the passive master would be derived by subtract one (–1) from bus parameter "TS" (This Station). For example, if TS in the database equals 2; the passive master would use address 1. If TS in the database equals 0 it would use address 125.

During switchover, when the passive master becomes active, the address would equal TS. Please note that this is handled by the Master module internally, the database downloaded to the two masters must be identical.

It is preferable to keep the address of the passive master as low as possible since it affects the time it takes to detect a faulty active master.

One issue regarding TS = 0 is that the passive master would attain address 125, which in turn means that HSA must be set to its maximum value of 125 to allow for the passive master to enter token ring. If this is not adhered to the Master module will reject mailbox END_INIT (Fault info = 6 and Additional Fault info = 19).

This special case would be handled by PCB master configuration which will not accept TS = 0 if the "HSBY" bus parameter profile is selected.

7.3.4 HSBY Master GSD-file

No changes to the original file PTQ_18F0.GSD would be necessary.

LED indicators

- The Master-status LED would flash red indicating that the Master module operates as passive master.
- When the Master module operates as passive master the COM-status LED would signal the same information as it does for an active master, or a combined C1/C2 master.

Unsupported functions

- When the Master module operates as a passive HSBY master the following mailboxes are not allowed and will be rejected with "Fault information code" 0x00FD.
  - FB_APPL_SET_SLAVE_MODE
  - FB_APPL_GET_MASTER_DIAG
  - FB_APPL_GET_SLAVE_DIAG (internal request)
  - FB_APPL_MSAC1_PROFID_V3_PARAM

- When the Master module operates as HSBY master (passive or active), "Class 2 master only" mode is not allowed. Refer to FB_INIT (special functions, bit 5).

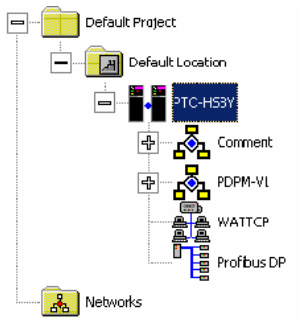
7.3.5 ProSoft Configuration Builder (PCB) HSBY Option Functionality

The PCB will be required to setup, control and identify matched Primary/Standby HSBY PTQ-PDPMV1 module sets.

Considerations should be identified for uploading and downloading of module configuration data as if the dual modules where a single module, where as upon downloading the configuration files, files are sent to both PTQ modules automatically.

The PCB will not be required to be concerned with Primary or Standby initially. However, the PCB will be required to know which module is Primary in order to perform accurate online monitoring. (Standby master modules will not be communicating with the slave devices)

A new icon is created to indicate the PROFIBUS configuration is in Hot Standby mode. The icon displays as a double PTQ module as seen in the following illustration.



PCB Master Configuration Software

The configuration software should readily make available HSBY diagnostic and status information in the buss view configuration mode. A separate and single master GSD file for a HSBY master should be created and used.

For simplicity, clarity and synchronization purposes the configuration software will allow for a single network configuration to be utilized for the HSBY PTQ master modules. For example, the user will configure a single network and download the configuration, the software will download the configuration files to both units and indicate the modules are synchronized and whether or not the download is successful for both modules.

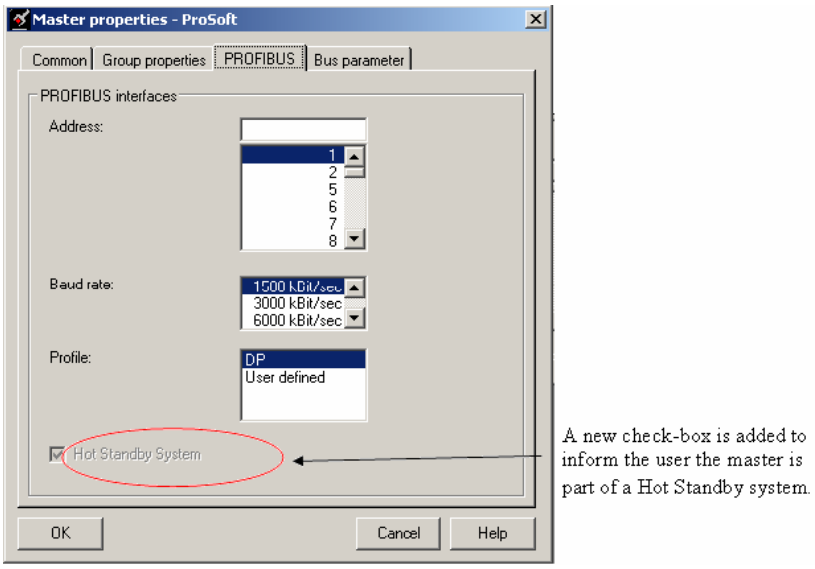
Downloading the modules configuration files will be transparent to the user. Both modules are required to be connected to the Ethernet network and upon download the PCB will automatically download configurations to both modules. If the download process is unsuccessful the PCB will inform the user and the

modules may have different CRC database value determined by the PROFIBUS FDL ping message.

The master(s) as they are configured will be initiated as neutral (as offline or not connected) until it reads the status information from the PLC. The master upon reading such status information will operate as Active, Passive or Offline mode.

Bus Parameters

The PROFIBUS master bus parameters for a Hot Standby project are automatically adjusted for best performance by the software. A new check-box in the PROFIBUS tab in the Master properties called "Hot Standby System" has been added to identify the master as a HSBY master set. The same Bus Configuration will be downloaded to both masters, ensuring that both the active and the passive master get exactly the same configuration and checksum values.



The Watchdog Time, Delta Ttr and other parameters can only be edited by selecting the User Defined profile.

When the "Hot Standby System" check-box is marked the PCB master configuration tool shall follow the rules below:

Parameter Calculation

Ttr is to be doubled in order to support one "passive" and one "active" Hot Standby master on the same network. Note that the Watchdog value also must be recalculated because of the doubled Ttr.

To allow a switch-over the Watchdog value that is calculated from the Ttr must be increased. Two new parameters are introduced: HsbyWatchdogFactor and Host Delay Time. The watchdog is calculated according to the following equation:

Watchdog = (Calculated watchdog * HsbyWatchdogFactor) + Host Delay Time

HsbyWatchdogFactor will be defined with separate values for each baud rate according to the table below:

Baud rate	XML keyword	Value
9.6 kBaud	HsbyWdFactor_9_6k	1
19.2 kBaud	HsbyWdFactor_19_2k	1
45.45 kBaud	HsbyWdFactor_45_45k	1
93.75 kBaud	HsbyWdFactor_93_75k	1
187.5 kBaud	HsbyWdFactor_187_5k	1
500 kBaud	HsbyWdFactor_500k	1
1.5 MBaud	HsbyWdFactor_1_5M	2
3 MBaud	HsbyWdFactor_3M	2
6 MBaud	HsbyWdFactor_6M	2
12 MBaud	HsbyWdFactor_12M	2

Host Delay Time is set to 300 (ms) in PCB Master Configuration. Station addresses.

Invalid Master addresses:

1 Master address must not equal 0.
2 Master address must not equal any assigned Slave address + 1.

Invalid Slave addresses:

1 A slave address must not equal the assigned Master address – 1.

7.4 PTQ-PDPMV1 Master Bus Properties when using a P&F DP/PA Segment Coupler

7.4.1 PROFIBUS DP time behavior

The Segment coupler supports both cyclic and acyclic data exchange of the PROFIBUS DP V1. Cyclic communication services are performed based on a specific time grid, which is referred to as the bus cycle time.

A bus cycle time (or cycle time for short) means the time lapse that goes by under "worst case" conditions to transfer the input data change of a PROFIBUS slave to the PROFIBUS master or the output data from the PROFIBUS master to the slave. All data that is to be updated cyclically is automatically updated in the common data area by the PROFIBUS PA master.

The cycle time, this actualization needs, depends on the data volume, that is transferred via the PROFIBUS PA channel. From the point of view of the PROFIBUS DP, the segment coupler represents a multi-slave. If the PROFIBUS DP Master sends a request to a slave address existing at the segment coupler, the gateway answers directly on the request with the data, that are stored in the common data range. Because of this behavior, the PROFIBUS DP master does not need to wait for the PROFIBUS PA slave to respond. The result of this is that the cycle time of the entire system is composed as follows:

```
tCycle = tCycle_PA-channel + tCycle_DP
```

The time tCycle_PA-channel can be estimated as follows:

```
tCycle_PA-channel = 10 ms + n*10.5 ms + 0.256 ms*(LE + LA)
```

where n = the number of PROFIBUS PA slaves

LE = total number of input bytes of all PROFIBUS PA slaves on the channel.

LA = total number of all output bytes of all PROFIBUS PA slaves on the channel.

The time tCycle_DP can be estimated as follows:

```
tcycle_DP = TBit * n * 500 + 11*TBit*(LE +LA)
```

where n = the number of PROFIBUS DP slaves

LE = total number of input bytes of all PROFIBUS slaves

LA = total number of output bytes of all PROFIBUS slaves

TBit = bit time = 1/transfer rate

For the time tCycle_DP a safety add-on of 10% should be included in the calculation in accordance with the PROFIBUS User Organization.

The equation above applies given the following pre-conditions:

- The PROFIBUS DP is operated as a mono-master system, i. e. there is only one master on the PROFIBUS DP. If you want to use a multi-master system, the token hold time and the corresponding pause times of the additional masters must be added to the total.
- Only acyclic data exchange takes place. If the master is also supposed to transfer acyclic telegrams, the time required for acyclic communication must be added in total.

7.4.2 Commissioning of communication with the SK1 segment coupler

Since the SK1 segment coupler works transparently, PROFIBUS PA stations are treated by the control system like PROFIBUS DP stations. This also applies to commissioning. To make it possible for the control system to exchange station data with a PROFIBUS, the GSD file of the station must be integrated into the configuration tool of the control system.

The SK1 segment couplers receive the PROFIBUS DP telegram, convert it simultaneously and transmit it on the PROFIBUS PA side. The PROFIBUS PA slave responds immediately to this telegram. The response telegram is received by the segment coupler, is again converted simultaneously and is transmitted on the PROFIBUS DP side as a slave response.

Note: To make it possible for data exchange between the PROFIBUS DP and PROFIBUS PA to work correctly, it is essential for the PROFIBUS DP transfer rate to be set to 93.75 kBd.

The time lapse between the master call and the slave response is limited. Since the PROFIBUS PA is working at a lower transfer rate than the PROFIBUS DP,

the standard settings of the bus parameters of the PROFIBUS DP Class 1 master must be changed.

Note: If the bus parameters are not changed, no data exchange is possible between the PROFIBUS DP master and the PROFIBUS PA slave.

PROFIBUS DP configuration tools do not always make it possible to set parameters for all bus parameters that are indicated in the following table. There are, however, dependencies that the configuration tool uses to calculate the dependant variable from the adjustable parameter value.

The following graph shows the standard settings of these parameter values for operation with the non-modular segment coupler:

Parameter	Value	Description
Baud rate [kBit/s]	93.75	PROFIBUS DP transfer rate
TSL[tbit DP]	4095	Slot-Time
Min TSDR [tbit DP]	22	Min. Station-Delay-Time
TID2 [tbit DP]	1000	Idle2-Time
Max TSDR	1000	Max. Station-Delay-Time
TID1 [tbit DP]	145	Idle1-Time
TSET [tbit DP]	55	Setup-Time
TQUI [tbit DP]	0	Quiet-Time
G	10	Gap-Factor
HSA	126	Highest-Station-Address
max_retry_limit	1	Repetitions in event of failure

The PROFIBUS DP transfer rate is fixed at 93.75 kbit/s for non-modular segment couplers. The bit time is thus tbit DP = 10.67 µs. The PROFIBUS PA transfer rate is fixed at 31.25 kBit/s; while the bit time is tbit PA = 32 µs.

The slot-time setting of the table above works if the total of input data bytes plus output data bytes < 253 bytes. If the data volume of a PROFIBUS PA slave exceeds this value, the slot-time should be set to 7192 [tbit DP]. For purposes of optimization, the ideal setting can be calculated as follows:

```
TSL > 13*(LS + LR) + 3*TSDR + 630
```

where LS is the number of data bytes in the Master_Request telegram and LR is the number of data bytes in the Slave_Response telegram.

Time TSDR refers to the actual time lapse that is counted between the Master_Request and the Slave_Response. This is typically 75 tbit DP.

Other possibilities for optimization in terms of cycle times are available through the Idle1-Time, Idle2-Time and the HSA setting.

The **Idle1-Time** TID1 is an idle time to be observed by the master between a response telegram and the prompt telegram following it. Parameters cannot not set directly for TID1 in many tools. To optimize the Idle-Time in spite of this, the Setup-Time TSET parameter or, if it cannot be adjusted either, the Quiet-Time TQUI parameter must be adjusted. The Idle-Time is calculated as follows:

```
TID1 = 2*TSET + TQUI + 35*tbit DP
```

The Idle1-Time TID1 depends on the maximum response time (not to be confused with the Station-Delay-Time TSDR) of all PA bus stations. In the table above, values are indicated for TID1 and TSET corresponding to the current PROFIBUS guidelines. In some circumstances, older PROFIBUS devices that do not yet work with response times in accordance with "PROFIBUS DP Expansion for EN 50170 (DPV1)", may provoke telegram repetitions. If this behavior occurs, you can increase Idle1-Time as an emergency measure. This will, however, increase the bus cycle time.

The **Idle2-Time** TID2 is the idle time between an SDN telegram (send data with no acknowledge) and the following call telegram. These SDN telegrams are used for global control services (SYNC, UNSYNC, FREEZE, UNFREEZE, and so on.). This value should be set to 1000 tbit DP.

If TID2 cannot be set directly, you can use the parameter max TSDR. If max TSDR is greater than TID1, as is shown in the table, the value of max TSDR is automatically used for TID2.

Highest-Station-Address HSA: A PROFIBUS master queries the status of all stations cyclically up to the address value HSA (1 telegram per cycle). As soon as one station at an address lower than HSA does not respond, (for example because it is not connected) the relatively long Slot-Time TSL expires until the next call telegram is transferred.

If it can be ensured that a station is present at every address including the HSA, this Slot-Time can be avoided.

7.4.3 Commissioning of communication with the SK2 segment coupler

Information for determining the Watch Dog Time TWD

PROFIBUS devices are able to activate a watchdog mechanism that monitors at each time interval of cyclic calls (data exchange) to make certain the PROFIBUS master is still active. The time measurement takes place in the PROFIBUS slave.

If the watchdog is activated and the time TWD (Watch Dog Time) since the last cyclic call expires, the device leaves cyclic data exchange, goes into the original state (Wait_prm) and sets the outputs to the secure state.

The value of the time TWD and the activation of the watch dog are transferred in the parameterized telegram from the PROFIBUS Master to the PROFIBUS slave at startup (transition to the data exchange). In general, dimensioning of the time TWD is used-specific (not device-specific, not in the GSD). The value is bounded below by cycle times.

As a rule, the configuration tool is used to enter the time TWD. There are configuration tools for which the watchdog time is set 1x per PROFIBUS master and other for which the watchdog time is set individually for each PROFIBUS PA station. This does not change with the value of the watchdog time.

For many tools, the time TWD is automatically calculated based on the cycle time of the master with a corresponding baud rate.

At higher baud rates on the PROFIBUS DP side (for example 12MBd) cycle times on the PA side may be longer by a factor of 300. If parameters were set directly for a PROFIBUS PA device at a baud rate (DP) for a time calculated for a higher master TWD, it would generally be less than the PA cycle and the device would not enter into the data exchange.

Behavior of segment coupler 2

To ensure reliable operation of the PROFIBUS the following bus parameters should be used:

- Transfer rate 45.45 kBd ... 12 MBd
- Watchdog time TWD = 5 s
- PROFIBUS DP Standard

Note: If there are a large number of PROFIBUS PA stations per channel of the SK2 segment coupler, the watchdog time TWD should be verified. The limit value is about 32 stations, but depends on the volume of data to be transferred.

Determining the parameter TWD

The watchdog time for the value above is a number based on experience, in other words one that usually works. If it turns out that the watchdog time is too long (slaves are not switching into the secure state quickly enough) or too short (slaves are switching into the secure state without the master ever having failed) this must be factored into the calculation. Depending on the configuration tool you are using, you can

- set parameters for only one watchdog time TWD for the entire PROFIBUS system. In this case, the greatest delay time must be used as the basis for determining TWD.
- set parameters for a watchdog time TWD for each individual slave.

The time TWD that is set (parameter) must be greater than the longest delay time TV_max that will occur. This is composed of a number of elements as follows:

```
TV_max = TCycle_DP + TCycle_PA_channel
```

where:

TCycle_PA_channel = the cycle time of the PROFIBUS PA channel

TCycle_DP = Cycle time of the PROFIBUS DP

Note: Pepperl+Fuchs recommends three times the PROFIBUS PA cycle time.

7.4.4 Details for calculating the TWD parameter

The PA cycle time TCycle_PA_channel depends on

1. The number n of stations on a channel
2. The effective data length LΣ (average of the total of input and output data of all devices [number of bytes (unit less)]):

Cycle time can be calculated in an approximate manner as

TCycle_PA_channel = n * (0.256ms * LΣ + 12ms) + 40ms

Note: For more information on calculating cycle time and other related data fro the SK1 or SK2 refer to PEPPERL & FUCHS, Instructions Manual Segment Coupler SK1 and SK2.

The above information references PEPPERL & FUCHS, Instructions Manual Segment Coupler SK1 and SK2. For additional information about the SK1 or SK2 please contact PEPPERL & FUCHS

8 Diagnostics and Troubleshooting

In This Chapter

- Basic Troubleshooting Steps 173
- LED Status Indicators - Front of PTQ module 174
- Module Status Indicators 176
- PROFIBUS Master Indicators 178
- The Configuration/Debug Menu 178

The module provides information on diagnostics and troubleshooting in the following forms:

- Status data values are transferred from the module to the processor.
- Data contained in the module can be viewed through the Configuration/Debug port attached to a terminal emulator.
- LED status indicators on the front of the module provide information on the module's status.

8.1 Basic Troubleshooting Steps

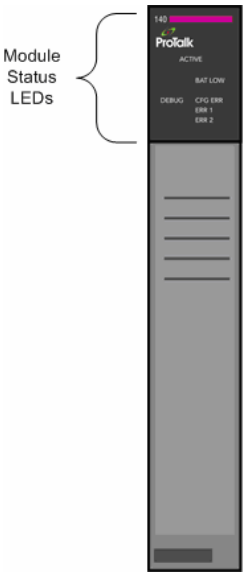
1. Verify that the module is installed correctly, and is communicating with the processor.
2. Install the most current version of ProSoft Configuration Builder.
3. Note the color and behavior of the LED Status Indicators (lights) on the front panel. Refer to the tables in the following section for examples.
   - The Active light should be ON whenever the module is communicating with the processor over the backplane. A processor must be positioned on the main rack and powered up. If this light does not come ON, either the module or the processor may be hung or off-line. Power cycle the PTQ and the PLC processor.
   - The Configure light should be OFF. If the light is ON, it is still possible to communicate with the module, but the module and master are prevented from going into RUN mode. There are two conditions that cause the Configuration light to be on:
     First: The Configuration files are missing or corrupt. Configuration files are stored on the Compact Flash card inserted in the rear of the module. Remove the Compact Flash card and transfer the configuration files from your PC to the card using a card reader.
     Second: The checksum values for PROFIBUS network configuration file have changed, but the processor did not acknowledge the configuration change by returning the correct checksum values to the module. Re-import the function block file using the procedure in the following section.

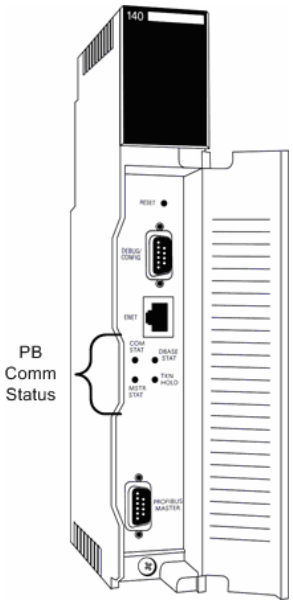
Special Note: Transferring Configuration Data to Replacement Module. All module configuration data (including PTQ, PROFIBUS Network, and Ethernet) are stored on the Compact Flash in the PTQ module. Should a module failure occur, it is a simple matter of moving the Compact Flash from the old module to the replacement module in order to transfer the configuration data.

8.2 LED Status Indicators - Front of PTQ module

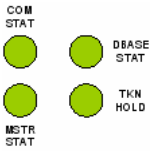
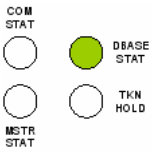
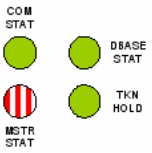
The LEDs indicate the module's operating status. The module has two sets of LED status indicators:

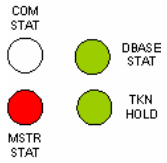
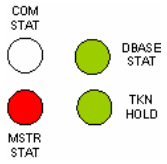
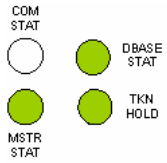
- PTQ Module Status LEDs on the front of the module near the top
- PROFIBUS Master Status LEDs behind the door on the front of the module.





The following table shows some of the possible status indicators:

	Processor Status	PTQ Module Status	PROFIBUS Master Status	PROFIBUS Master Status LED Description
Normal Operation	RUN (ON)	ACTIVE (ON)		COM STAT (Green/Solid or Flash): Master is communicating with slaves (Green) or at least one (Blinking). DBASE STAT (Green): PROFIBUS has been configured. TKN HOLD (Green): PTQ is holding the PROFIBUS token. MSTR STAT (Green): Master is in operate mode.
PTQ Module does not see the processor	RUN (ON) or STOP	ACTIVE (OFF)		COM STAT (OFF): Master is not communicating with slaves. DBASE STAT (Green): PROFIBUS has been configured. TKN HOLD (OFF): Master does not have the token and is inactive MSTR STAT (OFF): Master is inactive
HSBY: After the hot swap of the module the PTQ module is correctly SET as the Passive Master.	RUN (ON)	ACTIVE (ON)		COM STAT (Green/Solid or Flash): Master is communicating with slaves (Green) or at least one (Blinking) DBASE STAT (Green): PROFIBUS has been configured. TKN HOLD (Green): PTQ is holding the PROFIBUS token. MSTR STAT (Red Blinking): Master is in Passive Master mode.

	Processor Status	PTQ Module Status	PROFIBUS Master Status	PROFIBUS Master Status LED Description
PTQ PROFIBUS Master is Stopped	RUN or STOP	ACTIVE		COM STAT (Off): Master is not communicating with configured slaves. DBASE STAT (Green): PROFIBUS has been configured. TKN HOLD (Green): PTQ is holding the PROFIBUS token. MSTR STAT (Red): Master is in STOP mode.
CPU is Stopped	STOP	ACTIVE		COM STAT (Off): Master is not communicating with configured slaves. DBASE STAT (Green): PROFIBUS has been configured. TKN HOLD (Green): PTQ is holding the PROFIBUS token. MSTR STAT (Red): Master is in STOP mode.
CPU is Running	RUN	ACTIVE		COM STAT (Off): Master is operating but there is not any communication with slaves, or PROFIBUS Cable is disconnected. DBASE STAT (Green): PROFIBUS has been configured. TKN HOLD (Green): PTQ is holding the PROFIBUS token. MSTR STAT (Green): Master is in operate mode.

8.3 Module Status Indicators

Indicator	Color	Status	Indication
DEBUG	Green	On	Configuration/Debug Port is active
		Off	Configuration/Debug Port is inactive
CFG ERR	Red	On	Configuration Error - This LED is illuminated when the PROFIBUS and module CRC values do not match between input/output blocks. The module expects that the correct CRC values will be copied from the processor to the module, otherwise the module will be placed in STOP mode (MSTR STAT LED = RED) and the CFG ERR LED is illuminated to warn the user. Verify that the values match the values generated with the Calculate Checksums button in ProSoft Configuration Builder. A function block is provided for Concept to synchronize input CRCs with Output CRCs. This LED will also be illuminated if one of the module's configuration files is missing. This LED will illuminate if the module is placed in a slot other than the one for which it was configured. For HSBY, the LED will illuminate if the Active and Passive masters configuration files do not match. After downloading new configuration file to the PTQ module For more information on interpreting this error, refer to Word Offset 59 in *Slave List Structure*.
		Blink	Major error occurred. Please contact ProSoft Technical Support. This error is typically caused by a hardware incompatibility after firmware upgrade.
		Off	Normal Operation (configuration OK)
ERR1 and ERR2		On	The HSBY processors tried to perform a switch while a master was set to STOP mode. *This is a critical error and the module can be soft booted from within PCB diagnostics or the module may be reseated.*
Active	Green	On	The LED is on when the module is able to communicate over the backplane.
		Off	The LED is off when the module is unable to speak with the processor. The processor is either absent or not running.
		Flashing	This LED Flashes on the passive Master
BAT Low	Red	Off	The battery voltage is OK and running.
		On	The battery voltage is low or the battery is not present. The battery LED will illuminate briefly upon the first installation of the module or if the unit has not had power for an extended period of time. This behavior is normal, however, should the LED come on in a working installation, please contact the factory.
E-Link	Green	On	The Ethernet port is connected to the TCP/IP network.
E-Data	Green	On	There is data being transferred through the Ethernet port.

8.4 PROFIBUS Master Indicators

LED	State	Description
MSTR STAT	Green	Operating mode
	Green-Flashing	Clear mode
	Red	Stop mode
	Off	Offline
DBASE STAT	Green	Database OK
	Green-Flashing	Database download in progress
	Red	Invalid database
	Off	No databases have been downloaded
COM STAT	Green	Data exchange with all configured slaves
	Green-Flashing	Data Exchange with at least one of the configured slaves
	Red	Bus control error (possible bus short circuit or configuration error)
	Off	No data exchange with any configured slave
TKN HLD	Green	The module has the token
	Off	The module does not have the token.
ALL LEDs	Red	Fatal Error
	Off	HSBY Processor is stopped and master is held in reset state (inactive) The module is not in the configured slot.

8.5 The Configuration/Debug Menu

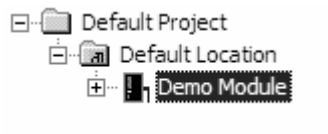
The Configuration and Debug menu for this module is arranged as a tree structure, with the Main Menu at the top of the tree, and one or more sub-menus for each menu command. The first menu you see when you connect to the module is the Main menu.

Because this is a text-based menu system, you enter commands by typing the command letter from your computer keyboard in the diagnostic window in ProSoft Configuration Builder (PCB). The module does not respond to mouse movements or clicks. The command executes as soon as you press the command letter — you do not need to press **[Enter]**. When you type a command letter, a new screen will be displayed in your terminal application.

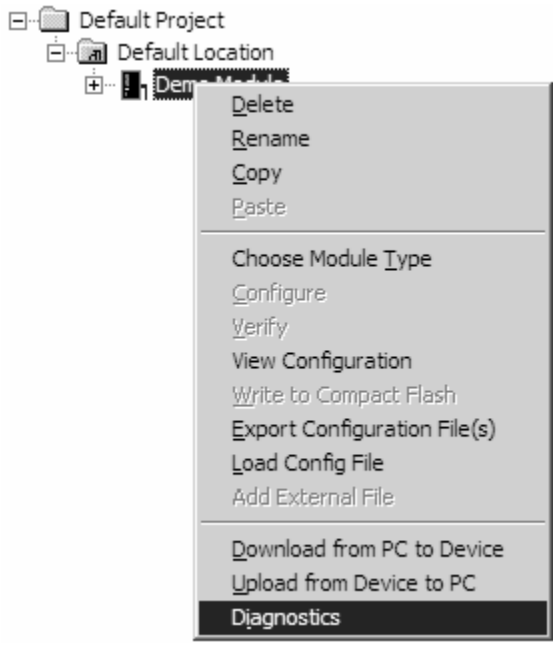
8.5.1 Using the Diagnostic Window in ProSoft Configuration Builder

To connect to the module's Configuration/Debug serial or Ethernet port:

1 Start PCB program with the application file to be tested. Right click over the module icon.



2 On the shortcut menu, choose Diagnostics.



3 This action opens the following dialog box. Press "?" to display the Main Menu.

```
PTQ-PDPMV1 COMMUNICATION MODULE MENU
  ?=Display Menu
  B=Block Transfer Statistics
  C=Module Configuration
  I=Input Data View
  O=Output Data View
  V=Version Information
  1=Module Status
  2=Fieldbus Data
  3=Control Registers
  4=Ethernet NIC Configuration
  @=View Network Configuration
  Esc=Exit Program and Reboot Module
```

If there is no response from the module, follow these steps:

1 Verify that the cable is connected properly between your computer's serial or Ethernet port and the module. A regular serial cable will not work.

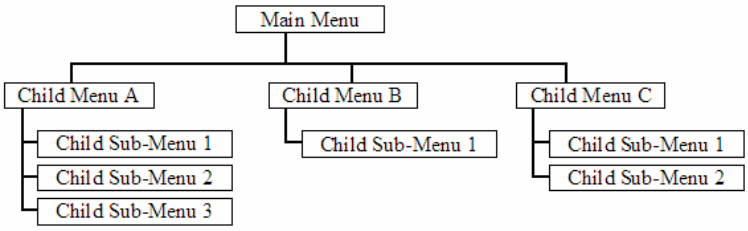
2 On computers with more than one serial port, verify that your communication program is connected to the same port that is connected to the module.

If you are still not able to establish a connection, you can contact ProSoft Technical Services for assistance.

8.5.2 Navigation

All of the sub-menus for this module contain commands to redisplay the menu or return to the previous menu. You can always return from a sub-menu to the next higher menu by pressing **[M]** on your keyboard.

The organization of the menu structure is represented in simplified form in the following illustration:



The remainder of this section shows you the menus available for this module, and briefly discusses the commands available to you.

Keystrokes

The keyboard commands on these menus are almost always non-case sensitive. You can enter most commands in lower case or capital letters.

The menus use a few special characters (**[?]**, **[-]**, **[+]**, **[@]**) that must be entered exactly as shown. Some of these characters will require you to use the **[Shift]**, **[Ctrl]** or **[Alt]** keys to enter them correctly. For example, on US English keyboards, enter the **[?]** command as **[Shift][/]**.

Also, take care to distinguish capital letter **[I]** from lower case letter **[l]** (L) and number **[1]**; likewise for capital letter **[O]** and number **[0]**. Although these characters look nearly the same on the screen, they perform different actions on the module.

8.5.3 Main Menu

When you first connect to the module from your computer, your terminal screen will be blank. To activate the main menu, press the **[M]** key on your computer's keyboard. If the module is connected properly, the following menu will appear on your terminal screen:

```
PTQ-PDPMV1 COMMUNICATION MODULE MENU
  ?=Display Menu
  B=Block Transfer Statistics
  C=Module Configuration
  I=Input Data View
  O=Output Data View
  V=Version Information
  1=Module Status
  2=Fieldbus Data
  3=Control Registers
  4=Ethernet NIC Configuration
  @=View Network Configuration
  Esc=Exit Program and Reboot Module
```

Caution: Some of the commands available to you from this menu are designed for advanced debugging and system testing only, and can cause the module to stop communicating with the processor or with other devices,

resulting in potential data loss or other failures. Only use these commands if you are specifically directed to do so by ProSoft Technology Technical Support staff. Some of these command keys are not listed on the menu, but are active nevertheless. Please be careful when pressing keys so that you do not accidentally execute an unwanted command.

Redisplaying the Menu

Press **[?]** to display the current menu. Use this command when you are looking at a screen of data, and want to view the menu choices available to you.

Viewing Block Transfer Statistics

Press **[B]** from the Main Menu to view the Block Transfer Statistics screen.

Use this command to display the configuration and statistics of the backplane data transfer operations between the module and the processor. The information on this screen can help determine if there are communication problems between the processor and the module.

```
BACKPLANE STATISTICS:

DATA TRANSFER CONFIGURATION:
  I/O CONFIGIGURATION:
    0x Bit Count   : 256      0x Word Count : 16
    1x Bit Count   : 256      1x Word Count : 16
    3x Word Count  : 1024     4x Word Count : 1024
    Battery Status : GOOD
  BLOCK COUNTS:
    Read  : 12333   Write : 12333   Error   : 0

HOT-STANDBY:
  Value   = 800E (%SW61 in Hex) [1000 0000 0000 1110]
  Current = 7 (Active Master)
```

Tip: Repeat this command at one-second intervals to determine the number of blocks transferred each second. If the module is communicating over the backplane correctly, you will see these block numbers change each time you refresh the display.

HSBY Note: The Quantum 140CPU67160 processor control word %SW61 is displayed for HSBY processor status

Viewing Module Configuration

Press **[C]** to view the Module Configuration screen.

Use this command to display the current configuration and statistics for the module.

Opening the Input Data View Menu

Press **[I]** to open the Input Data View Menu. Use this command to view the contents of the input database. Refer to Input Data View Menu for information on the commands on this menu.

Opening the Output Data View Menu

Press **[O]** to open the Output Data View Menu. Use this command to view the contents of the input database. Refer to Output Data View Menu for information on the commands on this menu.

Viewing Version Information

Press **[V]** to view Version information for the module.

Use this command to view the current version of the software for the module, as well as other important values. You may be asked to provide this information when calling for technical support on the product.

Values at the bottom of the display are important in determining module operation. The Program Scan Counter value is incremented each time a module's program cycle is complete.

Tip: Repeat this command at one-second intervals to determine the frequency of program execution.

Viewing Fieldbus Data

Press **[2]** to view Fieldbus data. Use this command to view information related to the status of each slave in the PROFIBUS network, and to verify that each slave is configured (SLAVE CFG LIST), exchanging data with the master (TRANSFER LIST) and in diagnostic mode (SLAVE DIAG LIST).

You can also check the operation state of the module, where:

- 00 = Offline
- 40 = Stop
- 80 = Clear
- C0 = Operate

Viewing Module Status

Press **[1]** to view information about Module Status. Use this command to view status information about the module. This screen also contains useful information for mailbox troubleshooting:

- Scan count
- Mailbox counters
- Alarm counters
- Hot Standby status
- Number of acyclic read and write operations performed by the module.

You can also view the number of mailbox messages in the input and output queues, and the number of alarms in the alarm queue.

HSBY Note: The following status reports new HSBY information.

```
HSBY State        = 3       Remote PE Mstr = 0883  Local PE Mstr = 0892
HSBY UDP Message  = Len=12   Msg in Hex: 0883 FE8C078E DAD2D7E9
```

HSBY State:

0 = Not Connected
1 = Passive
2 = Active
3 = Stand alone. This is not a HSBY state. Rather, it provides the state of the master when HSBY is disabled.

Remote PB Mstr:

Same as offset word 60
Low Byte: HSBY Passive Master Status
High Byte: HSBY Passive number of slaves

High Byte		Low Byte							
16	8	7				0			
		COM	--	--	OD	DB	CE	SO	PA

Bit	Explanation
PA	This bit indicates the state of the local master. 0 – Active master. that is, master is controlled by the Primary PLC. 1 – Passive master. that is, master controlled by the Stand-by PLC.
SO	This bit indicates if the local master recognizes any of its assigned slaves as "offline". 0 – At least one slave is "offline" 1 – All slaves OK
CE	This bit indicates if the local master has recognized a critical error. 0 – No critical errors 1 – Critical error active This bit is set when problems with the ping sequence is encountered.
DB	This bit indicates if the local master has detected a database mismatch. 0 – Database OK 1 – Database mismatch
OD	This bit indicates when the data in the Output data area of the DPRAM is updated after a switch over. 0 – Output data is not updated 1 – Output data is updated (Once this bit is set, it remains set for the remaining session until the Anybus is either reset or HSBY state changes to "Not Connected")
-	Not used set to zero
-	Not used set to zero
COM	This bit indicates if the counterpart is present. 0 = Counterpart not present 1 = Counterpart is present

Local PB Mstr:

Same as offset word 61
Low Byte: HSBY Active Master Status
High Byte: HSBY Active number of slaves

High Byte		Low Byte							
16	8	7				0			
		HS	-	-	OD	DB	CE	SO	PA

Bit	Explanation
PA	Bits 0 to 6 refer to Remote Byte reference above.
SO	
CE	
DB	
OD	
-	
-	
HS	This bit indicates that the Hot standby functionality is enabled. 0 – HSBY disabled. Module operates as "stand alone" master or HSBY-state equals "Not connected". 1 – HSBY enabled

HSBY UDP Message:

Same as offset word 62. This is the UDP message length.

Msg in HEX:

Same as offset word 63 to 67. UDP Message containing passive low byte, passive high byte, passive CRC32 checksum for PROFIBUS Master configuration, and CRC32 checksum for the module configuration.

Viewing Control Registers

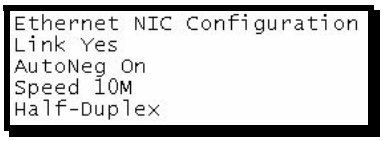
Press **[3]** to view information about the PROFIBUS Master's Control Registers. Use this command to view general information about the module, such as the firmware version and its serial number. The module status contains two possible codes:

- 0400 = module is running but not communicating with slaves,
- 0401 = running and communicating with slaves.

If the module is in STOP mode the status code will show as 0400.

Viewing Ethernet NIC Configuration

Press **[4]** to view the configuration for the Ethernet Network Interface Card (NIC) in the module.



Viewing the WATTCP.CFG File

Press **[@]** from the Network Menu. Use this command to view the module's IP address settings.

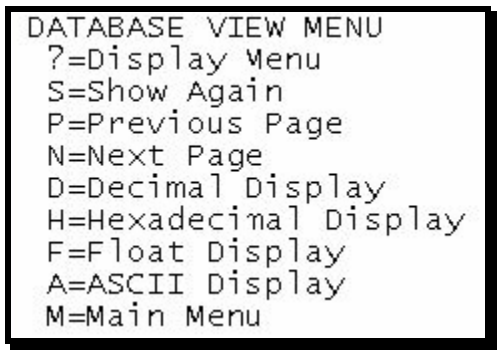
Exiting the Program

Caution: Some of the commands available to you from this menu are designed for advanced debugging and system testing only, and can cause the module to stop communicating with the processor or with other devices, resulting in potential data loss or other failures. Only use these commands if you are specifically directed to do so by ProSoft Technology Technical Support staff. Some of these command keys are not listed on the menu, but are active nevertheless. Please be careful when pressing keys so that you do not accidentally execute an unwanted command.

Press **[Esc]** to restart the module and force all drivers to be loaded. The module will use the configuration stored in the module's Flash ROM to configure the module.

8.5.4 Input Data View Menu

Use this menu command to view the current contents of the selected database. Press **[?]** to view a list of commands available on this menu.



Viewing the Previous Page of Data

Press **[P]** to display the previous page of data.

Viewing the Next Page of Data

Press **[N]** to display the next page of data.

Viewing Data in Decimal Format

Press **[D]** to display the data on the current page in decimal format.

Viewing Data in Hexadecimal Format

Press **[H]** to display the data on the current page in hexadecimal format.

Viewing Data in Floating Point Format

Press **[F]** from the Database View menu. Use this command to display the data on the current page in floating point format. The program assumes that the values are aligned on even register boundaries. If floating-point values are not aligned as such, they are not displayed properly.

Viewing Data in ASCII (Text) Format

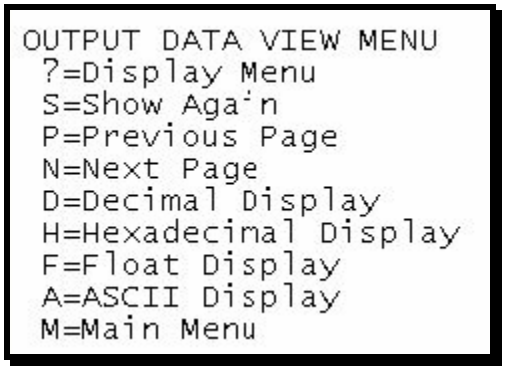
Press **[A]** to display the data on the current page in ASCII format. This is useful for regions of the database that contain ASCII data.

Returning to the Main Menu

Press **[M]** to return to the Main Menu.

8.5.5 Output Data View Menu

Use this menu command to view the current contents of the selected database. Press **[?]** to view a list of commands available on this menu.



Redisplaying the Menu

Press **[?]** to display the current menu. Use this command when you are looking at a screen of data, and want to view the menu choices available to you.

Viewing the Previous Page of Data

Press **[P]** to display the previous page of data.

Viewing the Next Page of Data

Press **[N]** to display the next page of data.

Viewing Data in Decimal Format

Press **[D]** to display the data on the current page in decimal format.

Viewing Data in Hexadecimal Format

Press **[H]** to display the data on the current page in hexadecimal format.

Viewing Data in Floating Point Format

Press **[F]** from the Database View menu. Use this command to display the data on the current page in floating point format. The program assumes that the values are aligned on even register boundaries. If floating-point values are not aligned as such, they are not displayed properly.

Viewing Data in ASCII (Text) Format

Press **[A]** to display the data on the current page in ASCII format. This is useful for regions of the database that contain ASCII data.

Returning to the Main Menu

Press **[M]** to return to the Main Menu.

9 Reference

In This Chapter

- Product Specifications 189
- Functional Overview 193
- PROFIBUS comDTM 206
- Cable Connections 216
- PROFIBUS Master Port 218
- Constructing a Bus Cable for PROFIBUS DP 218
- Supported PROFIBUS Services 222
- Quantum to PTQ Communication Protocol 223
- Calculating System Response Time 225
- Using Multiple PTQ-PDPMV1 Modules with Concept 228
- Frequently Asked Questions 229

9.1 Product Specifications

The PTQ-PDPMV1 module is a powerful communication interface for Quantum/Unity platform processors. Developed under license from Schneider Electric, the module incorporates proprietary backplane technology that enables powerful data exchange with Quantum or Unity processors.

The PTQ-PDPMV1 PROFIBUS DP/DPV1 Master module supports complete Master specifications according to IEC 61158. Acyclic parameter data can be transferred with Class 1 or Class 2 DPV1 services, allowing processors to easily communicate with slave devices supporting the PROFIBUS DPV0/V1 protocol.

The module now includes new features and functionalities supporting the Quantum 140CPU67160 Hot Standby processor in the Unity Pro XL programming environment.

The new (HSBY) Hot Standby features will be included with the current product offering, part number PTQ-PDPMV1.

Most PTQ-PDPMV1 modules firmware installed in the field can be flash upgraded, following the below guidelines and restrictions.

The module will operate identical to the stand-alone version with the following exceptions:

1 **Module Setup:** Once the module's firmware is updated, the module will automatically recognize the 140CPU67160 processor and activate the HSBY functions.
2 **PCB:** A new HSBY icon is displayed within PCB (ProSoft Configuration Builder) to identify the module as a HSBY unit.

Note: For specific HSBY instructions, pay attention to and follow the new HSBY icon for special instructions and guidance throughout this manual.

3 **Important Note on Field Firmware Flash Upgrades:** All modules having a serial number ≥ (greater than or equal to) 1451, having been shipped after 10/20/2005 can be field firmware upgraded. These modules have received a new hardware version 1.3 supporting the Hot Standby features. All other modules **must be** returned to the factory for firmware upgrades.

Caution: An attempt to field flash units with serial number 1450 or less require result in corrupt firmware image requiring the unit to be returned to the factory.

4 **Existing PROFIBUS networks:** PROFIBUS network baud rates greater than or equal to 500k baud are recommended to obtain a < 300ms switchover time, based on an average processor scan time of 100ms. For example, the switchover time for a network running 8 slaves utilizing 700 words input cyclic data and 700 words output cyclic data running at 500k baud with a processor scan time of 100ms is around 218ms. Please note Hot Standby units will increase the network token time as much as double because of two masters communicating on the network.
5 **Module configuration network:** The module requires Ethernet connectivity to operate properly. The modules use UDP messaging between each other to backup data in status registers used by the processor logic files to determine switchover conditions in the event PROFIBUS FDL ping messages fail (cut-cable).

The PROFIBUS DPV1 Master Module is a powerful communication interface for Quantum platform processors. Developed under license from Schneider Electric, the module incorporates proprietary backplane technology that enables powerful data exchange with Quantum processors.

The PROFIBUS DPV1 Master Communication Module is designed according to IEC 61158 specifications and has been tested and certified according to PTO guidelines to guarantee proper interoperability on a PROFIBUS network. Acyclic parameter data can be transferred with Class 1 or Class 2 DPV1 services, allowing processors to easily communicate with slave devices supporting PROFIBUS DPV0/V1 protocol.

9.1.1 Features and Benefits

- The module is built on Siemens ASIC ASPC2 Step E with Infineon C165 Microprocessor.
- The firmware is flash-upgradable.
- The PROFIBUS DPV1 module allows for slave messaging, diagnostics and alarm handling, and notification.
- Adheres to the Unity HEC Hot Standby Functional Specifications
- Cyclic Data: (Process I/O data) 1536 bytes of input and 1536 bytes of output data.
- Configure PTQ module and PROFIBUS network over Ethernet.

9.1.2 Hot Standby Support

The module provides support for 140CPU6716000 Hot Standby processor with UnityPro programming software. Look for the HSBY (Hot Standby) icon for special notes relating to the support and configuration of the module.

HSBY Note: For detailed understanding of HSBY specification, refer to the Hot Standby Addendum.

9.1.3 General Specifications

- Single Slot – Quantum backplane compatible
- The module is recognized as an Options module and has access to PLC memory for data transfer
- Configuration data is stored in non-volatile memory in the ProTalk module
- Configuration software for Microsoft Windows XP, 2000 and NT is included with the module.
- Up to six modules can be placed in a rack
- Local rack – The module must be placed in the same rack as processor.
- Compatible with common Quantum / Unity programming tools.
  - UnityPro XL
  - Concept
  - ProWORX
- Quantum data types supported: 3x, 4x
- High speed data transfer across backplane provides quick data update times.
- Sample ladder file available.

9.1.4 Hardware Specifications

Operating Temperature	0 to 60°C (32 to 140°F)
Storage Temperature	–40 to 85°C (–40 to 185°F)
Relative Humidity	5 to 95% (non-condensing)
LED Indicators	Module Status Backplane Transfer Status Serial Port Activity LED Serial Activity and Error LED Status Master Status Operations Network Drop Communication Master Token-Hold Master database configuration
Configuration Serial Port (PRT1)	DB-9M PC Compatible RS-232 only No hardware handshaking
Configuration Ethernet Port	RJ45 Connector Link and Activity LED indicators
PROFIBUS Master Port	DB-9F Optically Isolated RS-485 Ready, Run, Error and Token LED Indicators

9.1.5 Functional Specifications

- Easy-to-use drag and drop Busview configuration interface via ProSoft Configuration Builder software (see PSW-PCB Datasheet)
- Monitor/Modify of process data and DPV1 acyclic data with online slave diagnostics
- Supports PROFIBUS PA slaves on the network through DP/PA coupler or link
- Supports up to 125 slave devices with repeaters
- Supports extended diagnostic data (DPV1)
- Supports all standardized baud rates, up to 12 Mbits/s
- Auto baud detection at all valid PROFIBUS DPV1 rates
- Supports PROFIdrive 3.1 compliant parameter read and write operations
- Supports Sync and Freeze commands
- Alarm indications and confirmations handling (DPV1)
- Supports Multicast and Broadcast telegrams (DPV1)
- CRC checksum determination of slave configuration consistency to processor
- PROFIBUS DPV1 master comDTM software gateway included (see PSW-CDTM-PDPM) datasheet)

Hot Standby

- Hot Standby features supporting the SE 140 671 CPU.
- Supports a single PTQ-PDPMV1 module per rack.
- Diagnostic and status words are provided for Active Primary and Passive Secondary master health status.

- PROFIBUS switchover time will be nominal 100 ms not to exceed 300 ms.
- Cable break detection with segmented network slave quantity information.
- PROFIBUS health messages are generated from secondary master via FDL ping services.
- No setup parameters required. Module automatically detects Hot Standby system.

Physical

- PROFIBUS DPV1 RS-485 interface with a 9-pin D shell female connector and isolated Opto-Couplers
- Master Status LED Indicators for Operations, Network Drop Communication, Master Token-Hold

9.2 Functional Overview

9.2.1 About the PROFIBUS Protocol

PROFIBUS (Process Field Bus) is a widely-used, open-standards protocol created by a consortium of European factory automation suppliers in 1989.

PROFIBUS is a master/slave protocol. The master establishes a connection to the remote slave. When the connection is established, the master sends the PROFIBUS commands to the slave. The PTQ-PDPMV1 module works as a master only.

The PTQ-PDPMV1 module acts as an input/output module between devices on a PROFIBUS network and the Schneider Electric backplane. The module uses an internal database to pass data and mailbox requests and responses between the processor and the slave devices on the PROFIBUS network.

PROFIBUS supports a variety of network types. The network type supported by the PTQ-PDPMV1 module is PROFIBUS DP (Device Bus) version 1.0, which is designed for remote I/O systems, motor control centers, and variable speed drives.

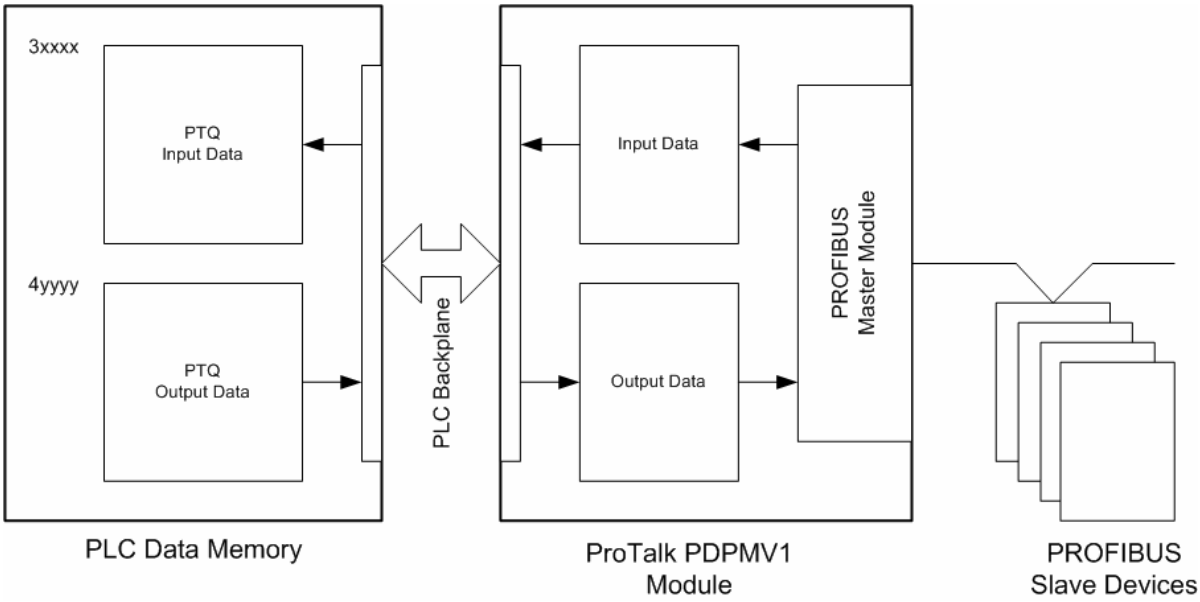
9.2.2 General Overview

The PTQ module communicates with the processor over the backplane using only the following two blocks of data:

- PTQ Input Data Block
- PTQ Output Data Block

This section of the manual describes the data structures and transfer mechanisms used to transfer data between the PTQ-PDPMV1 module and the Quantum processor.

The following illustration shows the Input/Output Data Block Flow between the Quantum processor and the PTQ-PDPMV1 Module.



These two data blocks (Input Data and Output Data) consist of a data structure that provides for the movement of:

- Input Data image from PROFIBUS Slave devices
- Output Data image for writing to PROFIBUS Slave devices
- PTQ Module Configuration and Status (from PTQ to Quantum)
- PROFIBUS Messaging Mailbox commands (from Quantum to PTQ)
- PROFIBUS Messaging Mailbox responses (from PTQ to Quantum)

9.2.3 PROFIBUS DP Architecture

The network supports multiple master systems with several slaves.

The following table shows the most important features of:

Standard	**EIN 501 70** **DIN 19245**
Transmission Equipment (Physical)	EIA RS-485 IEC 1158-2 (through link or coupler) Fiber Optic Cable (not available)
Transfer Procedure	Half-duplex
Bus Topology	Linear bus with active bus termination
Bus Cable Type	Shielded twisted pair conductors
Connector	9-pin D-Sub
Number of nodes on the bus	Max: 32 with no repeaters Max: 125 with 3 repeaters in 4 segments

Effective Range

Max Bus Cable Length Per Segment	Baud Rates (for 12 Mbit/sec cable)
1.2 km	9.6 kbit/sec
1.2 km	19.2 kbit/sec
1.2 km	93.75 kbit/sec
1.0 km	187.5 kbit/sec
0.5 km	500 kbit/sec
0.2 km	1.5 Mbit/sec
0.1 km	3 Mbit/sec
0.1 km	8 Mbit/sec
0.1 km	12 Mbit/sec

Bus Access

Two different bus access procedures handle the various communication requirements for the topology:

- Token Passing
- Polling

Token Passing

Token passing ring is the basis for communication between the more complex, active stations. All stations have the same rights in that a token is passed from station to station in a logical ring. The token is passed to each station with a maximum, definable token cycle time. A station is given transmission rights for the duration of time that it has the token.

Master/Slave Polling

Master/Slave polling guarantees a cyclic, real-time based data exchange between the station with transmission rights, active station and its subordinates, passive stations. In this case, the master is able to pass data to the slave and/or receive data. The services in layer 2 (field-bus data link in ISO-OSI reference model) organize this communication.

9.2.4 Master/Slave Communication Phases

The communication between the Master and the Slaves is split up into the following phases:

- Parameterization and configuration phase
- Usable data transfer phase

Before a DP Slave can be integrated into the usable data transfer phase, the parameterization and configuration phase runs a device identification test that verifies that the planned configuration matches the actual device configuration for each slave in the PROFIBUS network. The test verifies that:

- the device is actually there
- it is the right type of device

- the address, which is set on the device, matches the station address on the bus

  the formats, telegram length information, and bus parameters are correct and
- the number of configured inputs and outputs is correct.

9.2.5 PTQ Input and Output Data Blocks

The PTQ-PDPMV1 module reads PROFIBUS input data blocks from the Input Register range of data memory (3xxx for Quantum or %IWxxx for Unity). The PTQ-PDPMV1 module writes PROFIBUS output data to the Holding Register range of data memory (4xxx for Quantum or %MW for Unity). The size and starting register addresses of each data block is user configurable and must be specified during the setup of the module with ProSoft Configuration Builder.

Normal Operation

The PTQ-PDPMV1 module's application code initiates the data transfers at the end of every Quantum PLC ladder scan. As such, the PTQ-PDPMV1 module is able to actively read and write the PROFIBUS Cyclic Input/Output data blocks in the appropriate locations.

PTQ Input Data Block

Input/Output PTQ-PDPMV1 Image table

- Mailbox Messaging = Y
- Slave Diagnostics = N

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
1000 to 1159	%IW1000 to %IW1159	0 to 59	Configuration and Status data
1060	%IW1060	60	Low byte: HSBY Passive Status - from PROFIBUS interface High byte: HSBY Passive number of slaves - from PROFIBUS interface
1061	%IW1061	61	Low byte: HSBY Active Status - from PROFIBUS interface High byte: HSBY Active number of slaves - from PROFIBUS interface
1062	%IW1062	62	HSBY Message length - from UDP HSBY Server
1063	%IW1063	63	Low byte: HSBY Passive Status - from UDP HSBY Server Low byte: HSBY Passive number of slaves Message length - from UDP HSBY Server
1064 to 1165	%IW1064 to %IW1165	64 to 65	HSBY Standby PROFIBUS CRC32 - from UDP HSBY Server
1066 to 1167	%IW1066 to %IW1167	66 to 67	HSBY Standby User Cfg CRC32 - from UDP HSBY Server
1068 to 1172	%IW1068 to %IW1172	68 to 72	Reserved
1173	%IW1173	73	Number of Messages in the In Mailbox Queue
1174	%IW1174	74	Number of Messages in the Out Mailbox Queue

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
1175	%IW1175	75	Number of Messages in the Alarm Queue
1176	%IW1176	76	Last Out Mailbox Message ID processed from Output Image
1177	%IW1177	77	Current In Mailbox Control Index
1178	%IW1178	78	Current Alarm Control Index
1179 to 1222	%IW1179 to %IW1222	79 to 222	Incoming Mailbox Message data - 144 word Incoming Message block
1223 to N	%IW1223 to %IWN	223 to N	PROFIBUS Input Data - Data received from the PROFIBUS slaves devices on the network. Total size of the PROFIBUS Input data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (728 words). N is a function of the user-selected size of the PROFIBUS Input data block.

Input/Output PTQ-PDPMV1 Image with Mailbox messaging and slave Diagnostics table

- Mailbox Messaging = Y
- Slave Diagnostics = Y

Input Data Block (1369 words maximum)

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
1000 to 1172	%IW1000 to %IW1172	0 to 72	Configuration and Status data
1173	%IW1173	73	Number of Messages in the In Mailbox Queue
1174	%IW1174	74	Number of Messages in the Out Mailbox Queue
1175	%IW1175	75	Number of Messages in the Alarm Queue
1176	%IW1176	76	Last Out Mailbox Message ID processed from Output Image
1177	%IW1177	77	Current In Mailbox Control Index
1178	%IW1178	78	Current Alarm Control Index
1179 to 1222	%IW1179 to %IW1222	79 to 222	Incoming Mailbox Message data - 144 word Incoming Message block
1222 to 1600	%IW1223 to %IW1600	223-600	Incoming Slave 6 byte Diagnostics data for 126 slaves (378 words of data for slaves 0 to 125)
1601 to N	%IW1601 to %IWN	601 to N	PROFIBUS Input Data - Data received from the PROFIBUS slaves devices on the network.
			Total size of the PROFIBUS Input data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (728 words). N is a function of the user-selected size of the PROFIBUS Input data block.

Output Data Block (918 words maximum)

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
3000	%MW3000	0	Last in Mailbox Message ID
3001	%MW3001	1	Last Alarm Control index
3002 to 3003	%MW3002 to %MW3003	2 to 3	PROFIBUS CRC32 to Computed for PROFIBUS Configuration
3004 to 3005	%MW3004 to %MW3005	4 to 5	Module CRC32 - computed for module data When the module first starts up or recognized an initialization of the processor, it will compare the values of the two CRC's in the input and output images. If either one of the CRC's do not match, the module will be placed in Stop mode. If each set matches, the module will be placed in Operate mode.
3006 to 3149	%MW3006 to %MW3149	6 to 149	Outgoing Mailbox Data - Mailbox Message command being sent to the PTQ module
3150 to N	%MW3150 to N	150 to N	PROFIBUS Output Data - Data going to the PROFIBUS Network.
			Total size of the PROFIBUS Output data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (728 words). N is a function of the user-selected size of the PROFIBUS Output data block.

Input/Output PTQ-PDPMV1 Image without Mailbox messaging and with slave Diagnostics table

- Mailbox Messaging = N
- Slave Diagnostics = Y

PTQ Input Data Block without Mailbox Messaging and with Slave Diagnostics (1219 words maximum)

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
1000 to 1171	%IW1000 to %IW1171	0 to 71	Configuration and Status data
1172	%IW1172	72	Operating State Mode Return (New) Bit 15 = Handshake Bit Bit 14 =Error Bit (1=Error, 0=no error) Bits 12-13 = reserved for future use Bits 8-11 = Error code if bit 14 set: (1=Queue full, 2=Memory Allocation error, 3=invalid operating mode command) Bits 0-7 contain the operation code requested: 0x40= Stop 0x80= Clear 0xC0= Operate
1173 to 1450	%IW1173 to %IW1450	73-450	Incoming Slave 6 byte Diagnostics data for 126 slaves (378 words of data for slaves 0 to 125)

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
1451 to N	%IW1451 to %IWN	451 to N	PROFIBUS Input Data - Data received from the PROFIBUS slaves devices on the network.
			Total size of the PROFIBUS Input data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (728 words). N is a function of the user-selected size of the PROFIBUS Input data block.

PTQ Output Data Block without Mailbox Messaging and with Slave Diagnostics (774 words maximum)

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
3000	%MW3000	0	Set Operating Mode (New) Bit 15 = Handshake (if equal to Input word 72 Bit 15 then module has control of word and if not, then module has processed request) Bits 8-14= reserved for future use Bits 0-7 contain the operation code: 0x40= Stop 0x80= Clear 0xC0= Operate
3001	%MW3001	1	Reserved
3002 to 3003	%MW3002 to %MW3003	2 to 3	PROFIBUS CRC32 - Computed for PROFIBUS Config
3004 to 3005	%MW3004 to %MW3005	4 to 5	Module CRC32 - computed for module data When the module first starts up or recognized an initialization of the processor, it will compare the values of the two CRC's in the input and output images. If either one of the CRC's do not match, the module will be placed in Stop mode. If each set matches, the module will be placed in Operate mode.
3006 to N	%MW3006 to N	6 to N	PROFIBUS Output Data - Data going to the PROFIBUS Network.
			Total size of the PROFIBUS Output data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (728 words). N is a function of the user-selected size of the PROFIBUS Output data block.

Input/Output PTQ-PDPMV1 Image without Mailbox messaging and Without Slave Diagnostics table

- Mailbox Messaging = N
- Slave Diagnostics = N

PTQ Input Data Block without Mailbox Messaging or Slave Diagnostics (841 words maximum)

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
1000 to 1171	%IW1000 to %IW1171	0 to 71	Configuration and Status data
1172	%IW1172	72	Operating State Mode Return (New) Bit 15 = Handshake Bit Bit 14 =Error Bit (1=Error, 0=no error) Bits 12-13 = reserved for future use Bits 8-11 = Error code if bit 14 set: (1=Queue full, 2=Memory Allocation error, 3=invalid operating mode command) Bits 0-7 contain the operation code requested: 0x40= Stop 0x80= Clear 0xC0= Operate e
1173 to N	%IW1073 to %IWN	73 to N	PROFIBUS Input Data - Data received from the PROFIBUS slaves devices on the network.
			Total size of the PROFIBUS Input data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (728 words). N is a function of the user-selected size of the PROFIBUS Input data block.

PTQ Output Data Block without Mailbox Messaging or Slave Diagnostics (774 words maximum)

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
3000	%MW3000	0	Set Operating Mode (New) Bit 15 = Handshake (if equal to Input word 72 Bit 15 then module has control of word and if not, then module has processed request) Bits 8-14= reserved for future use Bits 0-7 contain the operation code: 0x40= Stop 0x80= Clear 0xC0= Operate
3001	%MW3001	1	Reserved
3002 to 3003	%MW3002 to %MW3003	2 to 3	PROFIBUS CRC32 - Computed for PROFIBUS Configuration
3004 to 3005	%MW3004 to %MW3005	4 to 5	Module CRC32 - computed for module data When the module first starts up or recognized an initialization of the processor, it will compare the values of the two CRC's in the input and output images. If either one of the CRC's do not match, the module will be placed in Stop mode. If each set matches, the module will be placed in Operate mode.

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
3006 to N	%MW3006 to N	6 to N	PROFIBUS Output Data - Data going to the PROFIBUS Network.
			Total size of the PROFIBUS Output data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (728 words). N is a function of the user-selected size of the PROFIBUS Output data block.

Input Block Status Data Details

The PTQ-PDPMV1 module's Input Data block contains several types of data in addition to the PROFIBUS network Input data. Much of this data is useful for determining the operational status and the configuration of the module.

The types of data returned in the PROFIBUS Input Status area include:

1 Module Configuration data values
2 PTQ Software revision level
3 Key PROFIBUS configuration values
4 PROFIBUS Master hardware/software revision level
5 PTQ module statistics
6 Mailbox messaging control and status

The following table is received from the PTQ module during every PROFIBUS Input Data Block update.

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Name	Description
1000	%IW1000	0	Module ID string	Unique module 10-byte pattern. Text identifier containing "PTQ-PDPMV1" string for module verification
1005	%IW1005	5	Quantum Slot Number	Slot number in the rack in which the PTQ-PDPMV1 module is located. Value is selected during user configuration
1006	%IW1006	6	PROFIBUS Input Data size	The number of words of PROFIBUS Input data to transfer from the PROFIBUS Master to the processor within the Input Data blocks. Value is selected during user configuration
1007	%IW1007	7	PROFIBUS Output Data size	The number of words to transfer from the processor in the PROFIBUS Output space during the Output data transfer. Value is selected during user configuration
1008	%IW1008	8	Input Data Start Address	Starting 4xxxx Holding Register address for Input Data block in processor data memory. Value is selected during user configuration
1009	%IW1009	9	Output Data Start Address	Starting 4xxxx Holding Register address for Output Data block in processor data memory. Value is selected during user configuration
1010	%IW1010	10	Reserved	Reserved

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Name	Description
1011	%IW1011	11H	Input Data Byte Swap	User configured flag to indicate if input data is swapped before placing in the input image for the controller. If 0, no swapping. If not 0, then swap bytes.
		11L	Output Data Byte Swap	User configured flag to indicate if output data is swapped after receiving from the output image of the controller. If 0, no swapping. If not 0, then swap bytes.
1012	%IW1012	12H	Module software major version number.	PTQ module application software version number.
		12L	Module software minor version number	Bits: 16 8 | 7 0 — High Byte | Low Byte
1013	%IW1013	13	PROFIBUS Slave Configured List	This is a 16-byte array with bit fields where one bit is assigned to each slave station address. The associate bit is set if the slave is present in the database. For the bit/slave relationship, refer to Slave List Structure.
1021	%IW1021	21	PROFIBUS Data Transfer Status	This is a 16-byte array with bit fields where one bit is assigned to each slave station address. The associated bit is set if the slave has reached or retained the "Data Exchange" state at least once during the last three data cycles. For the bit/slave relationship, refer to Slave List Structure.
1029	%IW1029	29	PROFIBUS Slave Diagnostic Status	This is a 16-byte array with bit fields where one bit is assigned to each slave station address. When a slave leaves the "Deactive" state for the first time the associated bit is set. This bit is then cleared when the slave enters "Data Exchange" state. If a slave indicates "Extended Diagnostics" when it is in "Data Exchange" stat, the associated bit is set. For the bit/slave relationship, refer to Slave List Structure.

Slave List Structure for Offset Words 13, 21 and 29

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Slave 7	Slave 6	Slave 5	Slave 4	Slave 3	Slave 2	Slave 1	Slave 0
Byte 1	Slave 15	Slave 14	Slave 13	Slave 12	Slave 11	Slave 10	Slave 9	Slave 8
Byte 2	Slave 23	Slave 22	Slave 21	Slave 20	Slave 19	Slave 18	Slave 17	Slave 16
Byte 3	Slave 31	Slave 30	Slave 29	Slave 28	Slave 27	Slave 26	Slave 25	Slave 24
Byte 4	Slave 39	Slave 38	Slave 37	Slave 36	Slave 35	Slave 34	Slave 33	Slave 32
...	...	...	...	...	...	...	...	...
Byte 15	Slave 127	Slave 126	Slave 125	Slave 124	Slave 123	Slave 122	Slave 121	Slave 120

Quantum Address (Example)	Unity Address (Example)	Word Offset	Name	Description
1037	%IW1037	37	PROFIBUS Master Operating State	PROFIBUS Master operating state 0x0000=Offline 0x4000=Stop 0x8000=Clear 0xC000=Operate
1038	%IW1038	38	PROFIBUS Ident Number	PROFIBUS Master PNO Ident number. Bytes will be swapped
1039	%IW1039	39	PROFIBUS Master Serial Number	This is a unique 32-bit serial number for the PROFIBUS Master.
1040	%IW1040	40	Reserved	Reserved for future use
1041	%IW1041	41	PROFIBUS Software Version	This is the software version number for the PROFIBUS Master software. Example for Version 1.40 High Byte - 0x40 Low Byte - 0x01
1042	%IW1042	42	PROFIBUS Master Module Status	Represents the PROFIBUS Master module's operating status. Bit 2=Application Status 0 - Application Stopped 1 - Application Running Bit 8=data exchange (FBRS) 0 - There is no data exchange with any of the assigned slaves 1 - There is Data Exchange with at least one of the assigned slaves Bit 9=slave input frozen/cleared (FBFC) 0 - A slaves inputs in the IN area are cleared in a slave is not in Data Exchange 1 - A slave's inputs in the IN area are frozen if a slave is not in Data Exchange Bit 12=reset (RDR) 0 - No action 1 - A reset is requested by the PROFIBUS Master module because a new database has been downloaded
1043 to 1044	%IW1043 to %IW1044	43	PROFIBUS configuration Checksum	CRC32 checksum for PROFIBUS Master configuration downloaded from configuration utility
1045 to 1046	%IW1045 to %IW1046	45	PTQ Module configuration checksum	PTQ-PDPMV1 module configuration checksum for module configuration downloaded from configuration utility
1047	%IW1047	47	Application program scan counter	PTQ module program scan counter. Can be used to gauge application code scan time performance
1048	%IW1048	48	Module PROFIBUS output image data update counter	Counter representing the number of times the output data image is transferred to the modules internal master.
1049	%IW1049	49	Module PROFIBUS input image data update counter	Counter representing the number of times the input data image is transferred from the modules internal master.
1050	%IW1050	50	Module out mailbox counter	Incremented at every mailbox requested from the module.

Quantum Address (Example)	Unity Address (Example)	Word Offset	Name	Description
1051	%IW1051	51	Module in mailbox counter	Incremented at every mailbox response sent to the Quantum
1052	%IW1052	52	Module alarm IND receive counter	Number of spontaneous alarm messages received from slave.
1053	%IW1053	53	Module alarm CON receive counter	Number of confirmation messages received from slaves indicating that the slave received the confirmation message from the PTQ module
1054	%IW1054	54	Reserved	
1055	%IW1055	55	Reserved	
1056	%IW1056	56	Module backplane read count	Rollover counter of the number of PTQ to processor backplane read data transfers
1057	%IW1057	57	Module backplane write count	Rollover counter of the number of PTQ to processor backplane write data transfers
1058	%IW1058	58	Module backplane error count	Rollover counter of the number of PTQ to processor backplane data transfers that have failed
1059	%IW1059	59	File Error Word	This is bit-mapped value that indicates which files are not present. Bit 0: Problem with PDPMV1.CFG file Bit 1: Problem with WATTCP.CFG file Bit 2: Problem with PDPMV1.DDB file Bit 3: Problem with PDPMV1.ZIP file Bits 4 to 15 not used. If this word has a value other than 0, the CFG ERR LED on the module will be illuminated.
1060	%IW1060	60	Low byte: HSBY Passive Status - from PROFIBUS interface High byte: HSBY Passive number of slaves - from PROFIBUS interface	

High Byte								Low Byte							
16				8				7				0			
								COM	--	--	OD	DB	CE	SO	PA

Bit	Explanation
PA	This bit indicates the state of the local master. 0 – Active master. that is, master is controlled by the Primary PLC. 1 – Passive master. that is, master controlled by the Stand-by PLC.
SO	This bit indicates if the local master recognizes any of its assigned slaves as "offline". 0 – At least one slave is "offline" 1 – All slaves OK
CE	This bit indicates if the local master has recognized a critical error. 0 – No critical errors 1 – Critical error active This bit is set when problems with the ping sequence is encountered.

<table>
<tr><th>Quantum Address (Example)</th><th>Unity Address (Example)</th><th>Word Offset</th><th>Name</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td></td><td>
<table>
<tr><td>DB</td><td>This bit indicates if the local master has detected a database mismatch.
0 – Database OK
1 – Database mismatch</td></tr>
<tr><td>OD</td><td>This bit indicates when the data in the Output data area of the DPRAM is updated after a switch over.
0 – Output data is not updated
1 – Output data is updated (Once this bit is set, it remains set for the remaining session until the Anybus is either reset or HSBY state changes to "Not Connected")</td></tr>
<tr><td>-</td><td>Not used set to zero</td></tr>
<tr><td>-</td><td>Not used set to zero</td></tr>
<tr><td>COM</td><td>This bit indicates if the counterpart is present.
0 = Counterpart not present
1 = Counterpart is present</td></tr>
</table>
</td></tr>
<tr><td>1061</td><td>%IW1061</td><td>61</td><td>Low byte: HSBY Active Status - from PROFIBUS interface
High byte: HSBY Active number of slaves - from PROFIBUS interface</td><td>
<table>
<tr><th colspan="8">High Byte</th><th colspan="8">Low Byte</th></tr>
<tr><td colspan="8">16</td><td colspan="8">8</td><td colspan="8">7</td><td colspan="8">0</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>HS</td><td>--</td><td>--</td><td>OD</td><td>DB</td><td>CE</td><td>SO</td><td>PA</td></tr>
</table>
<table>
<tr><td>HS</td><td>This bit indicates that the Hot standby functionality is enabled.
0 – HSBY disabled. Module operates as "stand alone" master or HSBY-state equals "Not connected".
1 – HSBY enabled</td></tr>
</table>
See explanation of bits OD, DB, CE, SO and PA in chart for address 1060.
</td></tr>
<tr><td>1062</td><td>%IW1062</td><td>62</td><td>HSBY Message length - from UDP HSBY Server</td><td></td></tr>
<tr><td>1063</td><td>%IW1063</td><td>63</td><td>Low byte: HSBY Passive Status - from UDP HSBY Server
High byte: HSBY Passive number of slaves Message length - from UDP HSBY Server</td><td>Refer to word 60 for explanation. This is a backup word derived from Ethernet UDP messaging</td></tr>
<tr><td>1064 to 1165</td><td>%IW1064 to %IW1165</td><td>64 to 65</td><td>HSBY Passive PROFIBUS CRC32 - from UDP HSBY Server</td><td>CRC32 checksum for PROFIBUS Master configuration downloaded from configuration utility via UDP</td></tr>
<tr><td>1066 to 1167</td><td>%IW1066 to %IW1167</td><td>66 to 67</td><td>HSBY Passive User Cfg CRC32 - from UDP HSBY Server</td><td>PTQ-PDPMV1 module configuration checksum for module configuration downloaded from configuration utility via UDP</td></tr>
</table>

Quantum Address (Example)	Unity Address (Example)	Word Offset	Name	Description
1067 to 1072	%IW1060 to %IW1072	67 to 72	Reserved	Reserved for future use
1073	%IW1073	73	In mailbox queue count	Number of message in the in mailbox queue
1074	%IW1074	74	Out mailbox queue count	Number of message in the out mailbox queue
1075	%IW1075	75	Alarm queue count	Number of message in the alarm queue
1076	%IW1076	76	Last out mailbox message ID processed from output image	The module confirms the receipt of a mailbox by copying its ID code (Message ID) to this register.
1077	%IW1077	77	Current in mailbox control index	Incremented after the module has transferred a new mailbox response to the processor.
1078	%IW1078	78	Current alarm control index	Incremented after the module has transferred a new alarm to the processor.

PTQ Output Data Block

Quantum Address (Example)	Unity Address (Example)	Relative Word Offset	Description
3000	%MW3000	0	Last in Mailbox Message ID
3001	%MW3001	1	Last Alarm Control index
3002 to 3003	%MW3002 to %MW3003	2	PROFIBUS CRC32 - Computed for PROFIBUS Config
3004 to 3005	%MW3004 to %MW3005	4	Module CRC32 - computed for module data When the module first starts up or recognized an initialization of the processor, it will compare the values of the two CRCs in the input and output images. If either one of the CRCs do not match, the module will be placed in Stop mode. If each set matches, the module will be placed in Operate mode.
3006 to 3149	%MW3006 to %MW3149	6 to 149	Outgoing Mailbox Data - Mailbox Message command being sent to the PTQ module
3150 to N	%MW3150 to N	150 to N	PROFIBUS Output Data - Data going to the PROFIBUS Network. Total size of the PROFIBUS Output data block is a function of the PROFIBUS network configuration. Maximum size is 1536 bytes (768 words). N is a function of the user-selected size of the PROFIBUS Output data block.

9.3 PROFIBUS comDTM

DTM (Device Type Manager) is a standard way to provide all necessary data and functionality for a communication device, for example a PROFIBUS DP card. This technology is similar to the way Microsoft Windows supports printer drivers supplied by manufacturers and available to any Windows application, rather than requiring a custom printer driver for each specific application.

PROFIBUS comDTM, distributed by ProSoft Technology, is a DTM for the PTQ-PDPMV1 module (Ethernet) and MVI series PDPMV1 modules (RS232 serial), which allows configuration tools, instruments and communication devices on a field network to recognize and use the module's capabilities.

9.3.1 ProSoft Technology product availability

Part Number	Description
PSW-cDTM-PDPM	PROFIBUS DPV1 master comDTM software gateway

9.3.2 Introduction

Use the following topics to introduce yourself to the product and answer questions you may have.

Why use PROFIBUS comDTM?

Customers from around the world have different machines, fieldbusses, and other end-user equipment. Each is equipped with the field bus requested by their end-user. Since there are so many variations, the automation solution in their standard machine ends up being different from case to case.

This means that service engineers need to have different configuration tools for every fieldbus. Or maybe even one for every device. You want one, but the reality is you have many. This is where PROFIBUS comDTM can help with configuring and communicating with different networks, different products and different suppliers.

What is FDT?

FDT (Field Device Tool) is the specification for software interfaces for the integration of field devices, drives, and controls in engineering and configuration tools. FDT is manufacturer independent and allows the user trouble-free parameterization and configuration of his specific processing system.

FDT technology standardizes the communication interface between field devices and systems. The key feature is its independence from the communication protocol and the software environment of either the device or the host system. FDT allows any device to be accessed from any host through any protocol.

The FDT container implements the FDT specification. It serves as an interface between FDT and a superior application. It uses the DTMs to gain access to the

devices. FDT frame application is an engineering or configuration tool which has and FDT container.

FDT technology comprises three key components: the Frame Application, Device DTMs, and Communication DTMs.

- The DTM (Device Type Manager) is used for the configuration and maintenance of field devices, drives etc. It is only functional with an FDT container.
- The FDT container implements the FDT specification. It serves as interface between FDT and a superior application. It uses the DTMs to gain access to devices.
- FDT frame application is an engineering or configuration tool which has an FDT container. The user interface of the DTMs is displayed here.

To better understand the functionality of these components, consider the analogy of the Internet -- a standard web browser allows users to view countless web pages created by many content providers. The host system supplier typically creates the Frame Application, just as Microsoft supplies the Internet Explorer web browser. Just as a web browser opens a web page that contains code from the company that makes the web page, an FDT frame opens the Device DTM, which is the plug-in software from the device vendor.

Similar to a web browser, the Frame Application has menu bars, toolbars, and a navigation tree. Visually, the frame application surrounds the device vendor's DTM. Like opening a web page from a 'favorites' navigation tree, a user can navigate down a tree that lists the field device tags, click on one, and open the device vendor's DTM inside the frame. And, like web pages that let users interact with a reservation system or a shopping service, the Device DTMs let the user interact with the field device in a wide variety of ways. The Device DTM vendor can create a graphically rich user interface that does virtually anything possible in an advanced Windows PC-type interface. The third part of the technology, the Communication DTM, provides a standardized communication Application Process Interface (API) inside the PC, interfacing between the Device Vendor's DTM and the host system's specific driver that handles pass-through communications from the PC down to the fieldbus interface card.

The host system vendor supplies a Communication DTM (comDTM) for each supported fieldbus protocol. This ensures that the details of the PC, network, interface cards, and pass-through protocols of the host system, are transparent to the device vendor's DTM. This correlates back to the internet analogy where: the web page is transparent to the PC it's running in, the brand of the network interface card in the PC, or whether communication is DSL or broadband cable.

FDT technology complements and expands, existing device description languages. It does not replace but rather builds upon existing DDs.

In particular, FDT expands the capabilities of DD for complex devices. Device Description languages have limitations in the graphical representation of the device at the user interface and allow only a limited integration of special features. FDT/DTM removes these limitations.

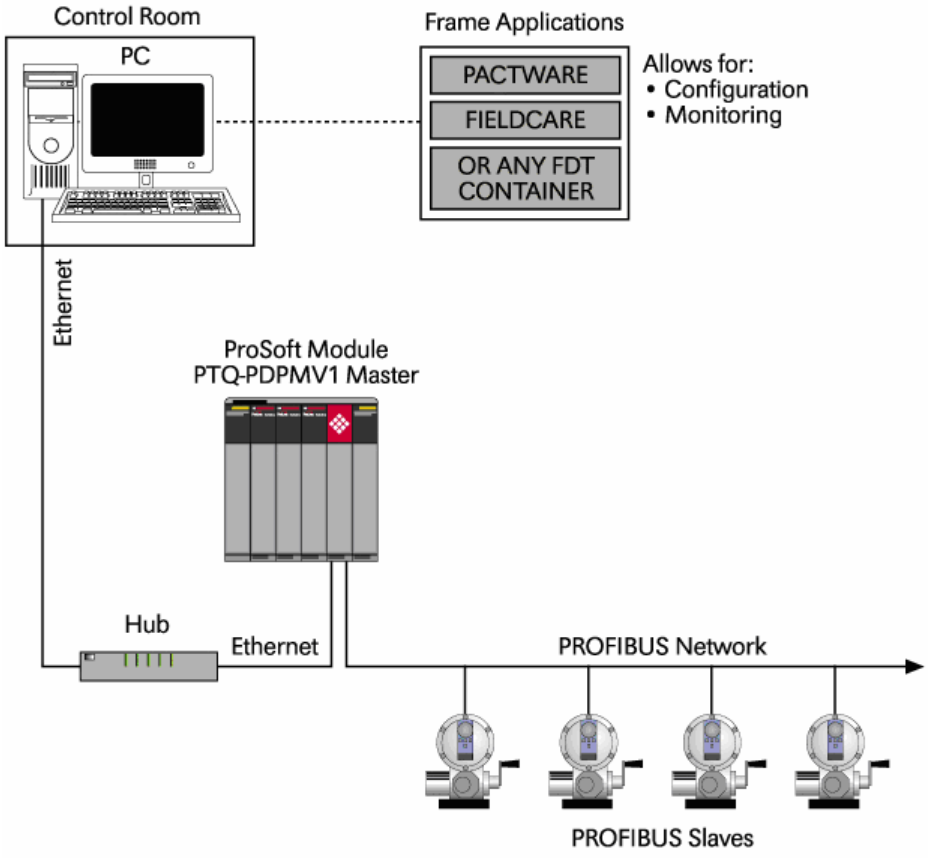
Typical frame applications are

- Pactware from The PACTware Consortium e.V. (freeware)
- FieldCare from Endress & Hauser
- Field Control from ABB

What is DTM?

DTM (Device Type Manager) is a standard way to provide all necessary data and functionality for a communication device, for example a PROFIBUS DP card. This technology is similar to the way Microsoft Windows supports printer drivers supplied by manufacturers and available to any Windows application, rather than requiring a custom printer driver for each specific application.

PROFIBUS comDTM, distributed by ProSoft Technology, is a DTM for the PTQ and MVI series PDPMV1 modules, which allows configuration tools, instruments and communication devices on a field network to recognize and use the module's capabilities.



What is PROFIBUS?

PROFIBUS (Process Field Bus) is a widely-used, open-standards protocol created by a consortium of European factory automation suppliers in 1989.

PROFIBUS is a master/slave protocol. The master establishes a connection to the remote slave. When the connection is established, the master sends the PROFIBUS commands to the slave.

PROFIBUS supports a variety of network types. The network type supported by the PTQ and MVI series PROFIBUS modules is PROFIBUS DP (Device Bus) version 1.0, which is designed for remote I/O systems, motor control centers, and variable speed drives.

System Requirements

Confirm that your system meets the following hardware and software requirements before you start the installation.

9.3.3 Hardware Requirements (Recommended)

- Pentium 4 Processor rated for at least 2 GHz
- 450 MB hard drive space for DTM Libraries
- Video card capable of 1024 X 768 resolution at 64k colors
- Ethernet Network Interface Card (NIC) (PTQ-PDPMV1 only)
- ProSoft Technology Profibus DPV1 Master module as follows:
  - PTQ-PDPMV1 Ethernet only, serial port not supported
  - MVI series PDPMV1 RS232 serial only.

Note for MVI series and PTQ Users: The Ethernet connection implements UDP protocol, which dynamically allocates a random UDP port for every connection. This implementation limits the possibility of using most serial to Ethernet converters to access the MVI series PDPMV1 serial port through an Ethernet connection. Several Ethernet to serial converters require the configuration of a fixed UDP port, which is not available for the current implementation.

9.3.4 Software Requirements (Minimum)

- Windows NT 4.0 Service Pack 6A, or Windows 2000 SP3 or Windows XP Professional SP2.
- Microsoft Internet Explorer Version 6.0
- FDT 1.2.1 compliant FDT frame application. Compatible applications include:
  - PACTware
  - FieldCare

MM Container Some FDT Containers require the following components:

- Microsoft Management Console
- Adobe Acrobat Reader 5.0

Installation

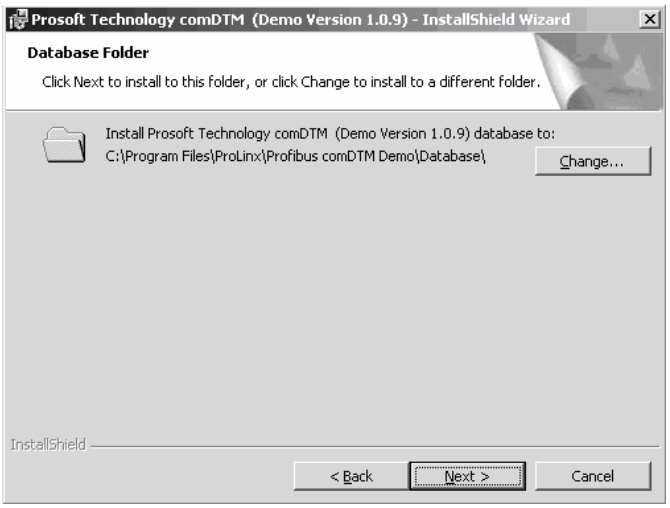
Important: You must have "Administrator" rights on your computer to install this application.

Important: Please open and read the release notes on the CD before starting the installation. The release notes are located in the root directory of the CD-ROM.

➢ *To install comDTM:*

1. Insert the comDTM CD-ROM in an available CD-ROM drive in your computer.
2. On most computers, the installation program will start automatically within a few seconds. If the installation does not start automatically on your computer, click the Start button, choose Run, and then type `explorer`. Click OK to start Windows Explorer. In Windows Explorer, open the My Computer icon and navigate to the CD-ROM drive. Double-click the file "Setup.exe". This action starts the installation wizard.
3. Follow the instructions on the installation wizard to install the program.
4. Click **Finish** to complete the installation. If you are prompted to restart your computer, save your work in any applications that are running, close the applications, and allow the computer to restart.

Note: During installation, you will be prompted to accept or change the location for the database folder. The default location for this folder is the Program Files directory on your local hard drive (normally Drive C:). If you intend to allow multiple workstations to access the same database folder, you should choose a network drive that other workstations can access.


Prosoft Technology comDTM (Demo Version 1.0.9) - InstallShield Wizard
Database Folder
Click Next to install to this folder, or click Change to install to a different folder.
Install Prosoft Technology comDTM (Demo Version 1.0.9) database to:
C:\Program Files\ProLinx\Profibus comDTM Demo\Database\
Change...
InstallShield
< Back
Next >
Cancel

Quick Start

The following steps demonstrate how to start the FDT (Field Device Tool) program and configure the PROFIBUS comDTM.

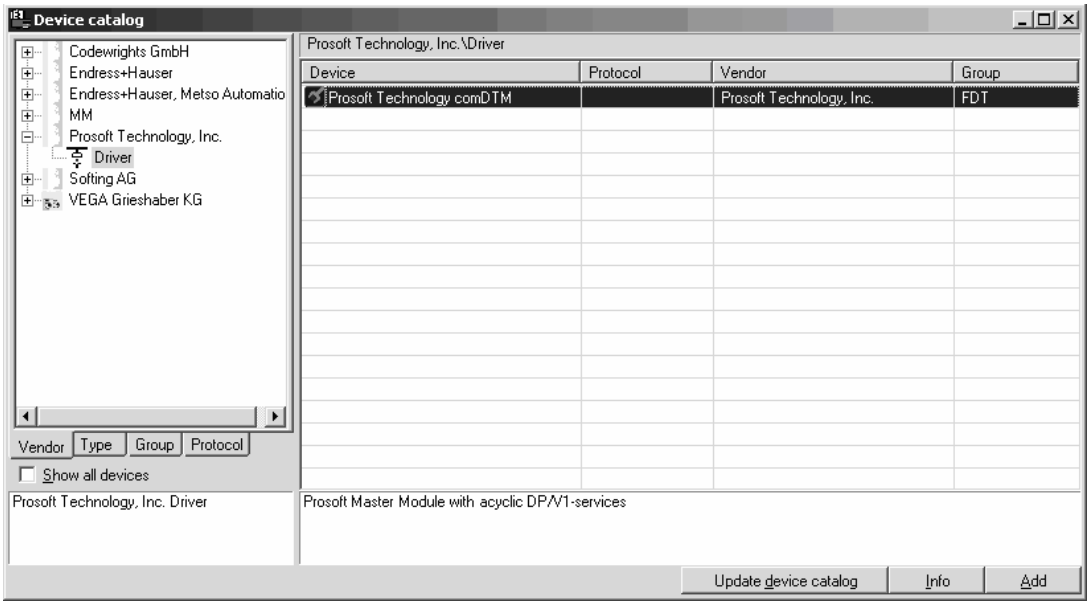
The comDTM provides a Guided Tour section in the Online Help that explains the basic features and operation of the program. To open the online help, click

the right mouse button on ProSoft Technology comDTM, and choose "Additional Functions - Online Help" from the shortcut menu.

➢ *Starting FDT*

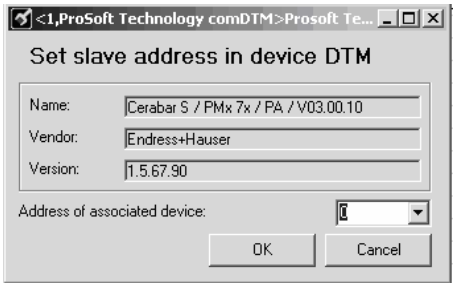
1 Start the FDT program and login as administrator. The following procedures use PACTware 3.0 software.

2 Update or refresh the device catalog. If PROFIBUS comDTM was installed successfully, ProSoft Technology will be listed in the Vendor Tab.

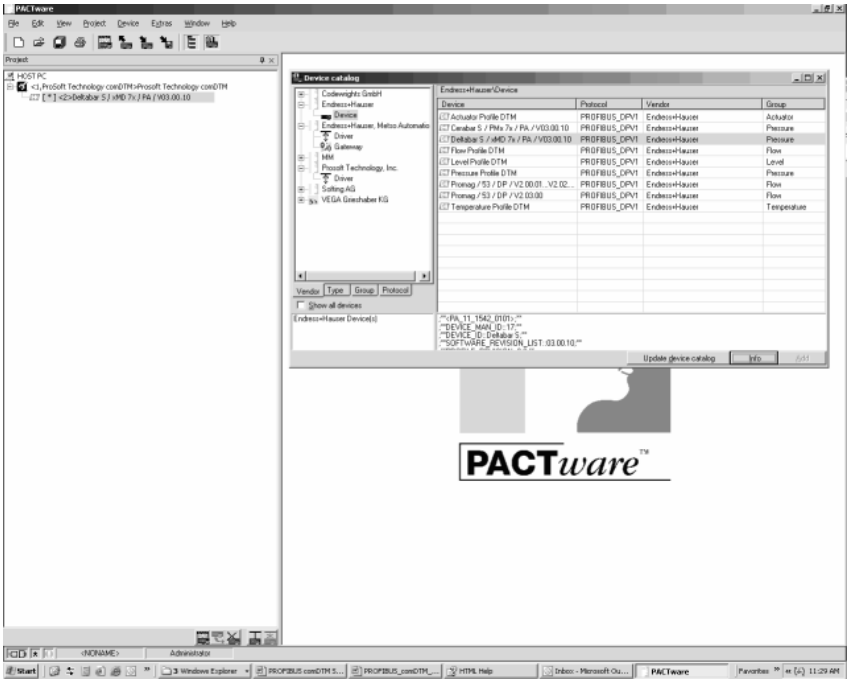


3 Select the ProSoft Technology icon, and then click "Add"

4 Follow steps 1 through 3 to add any other manufacturer's device DTMs installed on your computer. Select the correct address for each device, and then click "Ok" to proceed.

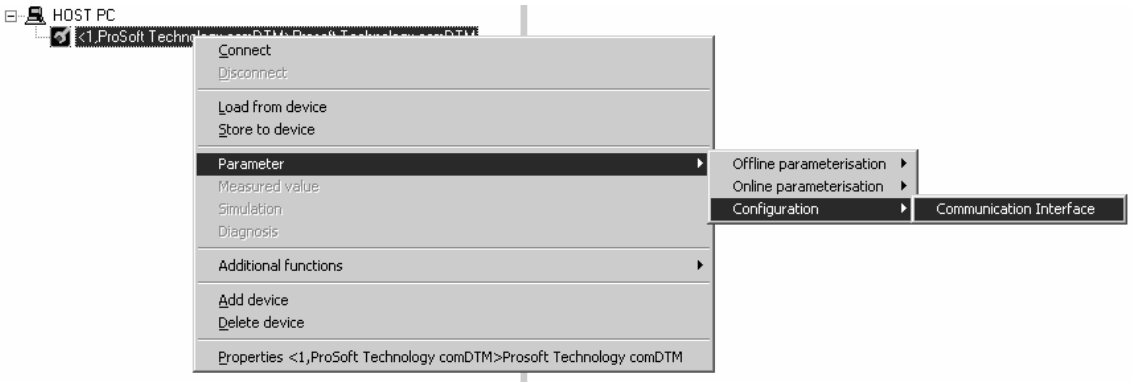


Note: You must use the same PROFIBUS device address that you used when setting up the device.

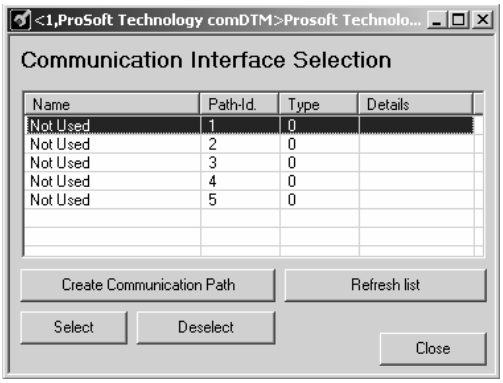


➢ *Connecting the comDTM to the master to establish communication*

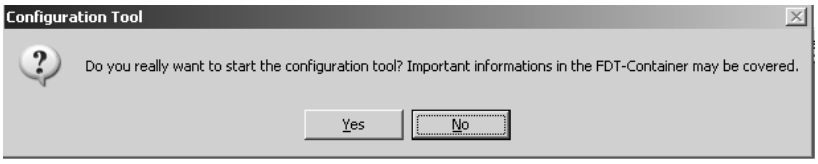
1 Select the comDTM icon and click the right mouse button to open a shortcut menu. On the shortcut menu, choose "Parameter", then "Configuration", then "Communication Interface".



2 If you have not created a communication path, select "Create Communication Path". If you have already created a path, skip this procedure and go to step 5.



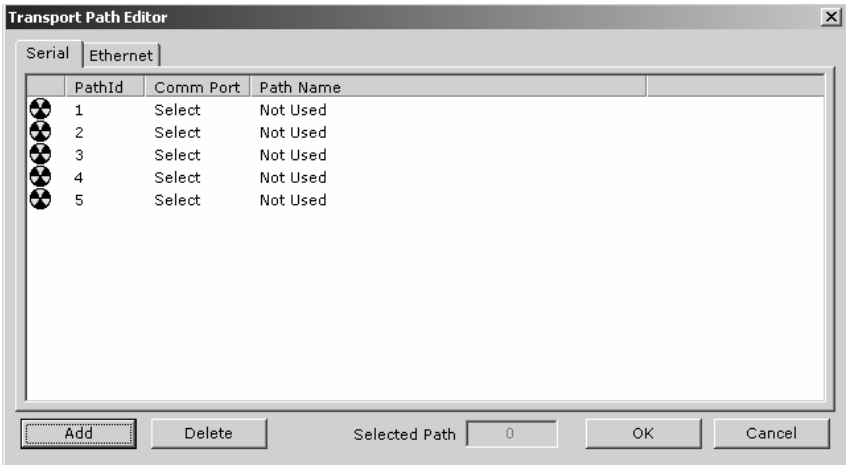
3 Click the "Configuration Tool" button, and then click "Yes" on the Configuration Tool dialog box to create a new path.


Configuration Tool
Do you really want to start the configuration tool? Important informations in the FDT-Container may be covered.
Yes
No

4 Create a new path.

a) **PTQ users**: Click the Ethernet tab and then click the "Add" button. Double click on the IP address and enter the IP address of the ProSoft Technology PROFIBUS master module IP address. Add a descriptive 'Path Name'.

b) **MVI users**: Click the Serial tab and then click the "Add" button. Select the serial COM number used when configuring the MVI module. Add a descriptive "Path Name".

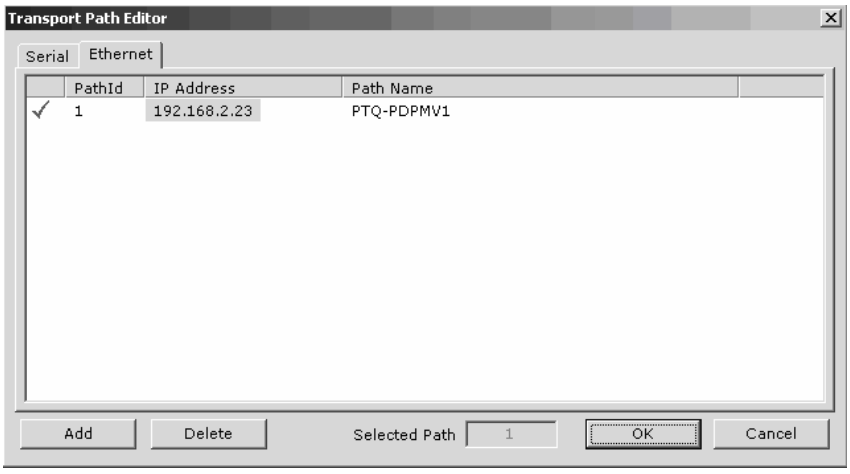


Note:

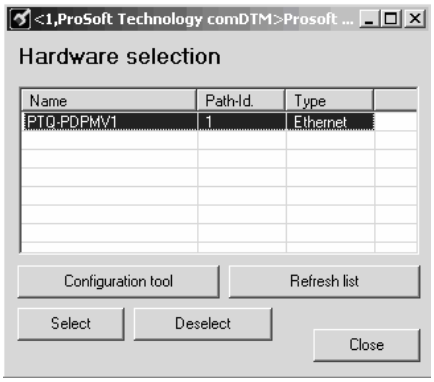
PTQ users: This is the IP address you selected when downloading configuration files using ProSoft Configuration Builder.

MVI users: This is the COM port selected when downloading configuration files using ProSoft Configuration Builder.

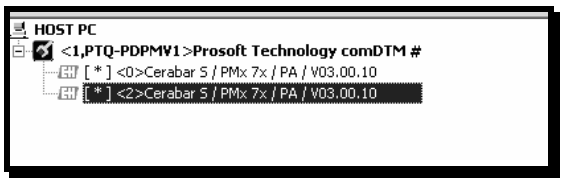
5 The Path Name is a "friendly name" for your reference. Enter a path name that identifies the module, and then click OK to return to the previous dialog box.



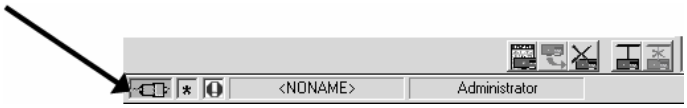
6 Select the new path and then click the "Select" button. Click the "Close" button. This action associates the device with the path ID in your field network.



7 Select the comDTM icon and click the right mouse button to open a shortcut menu. On the shortcut menu, choose "Connect". If the connection is successful, the icon will be highlighted, as shown in the following illustration:

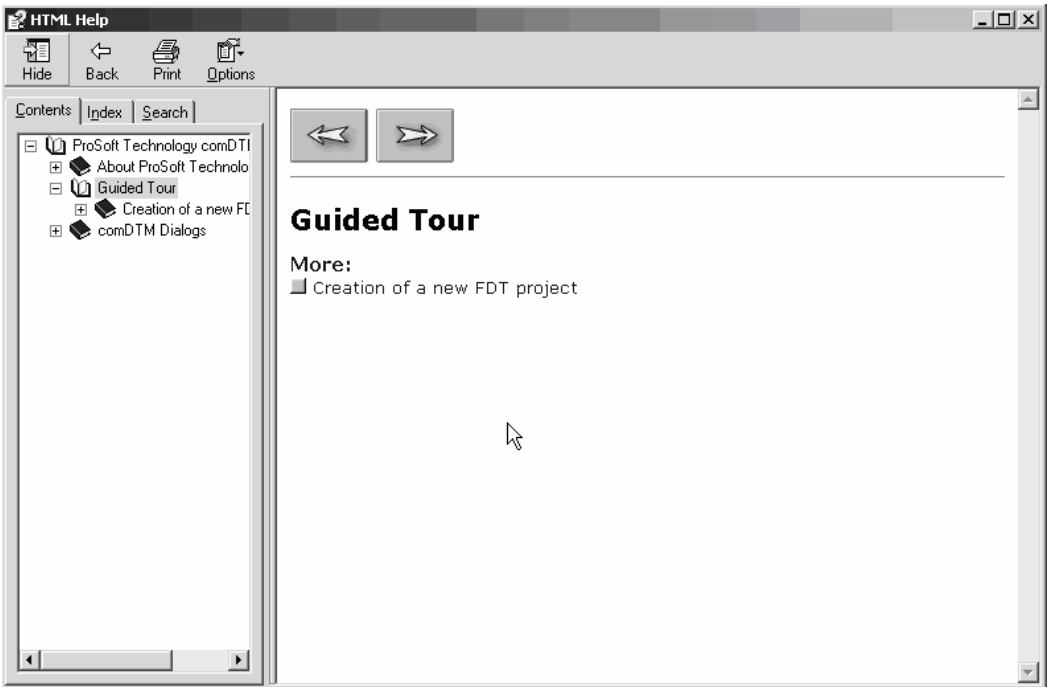


When the comDTM is connected with the master, PACTware indicates the connection master by displaying a green plug in the status bar.



This completes the installation and Quick Start Guide for the ProSoft Technology PROFIBUS comDTM. Refer to the online help and documentation additional information on each DTM component you have installed and configured.

To open the comDTM the help file, select the comDTM object and click the right mouse button to open a shortcut menu. On the shortcut menu, choose "Additional Functions", then "Online Help" and then open the "Guided Tour" icon. Use the navigation buttons in each help page to view the help topics.



Refer to the documentation and online help for your FDT frame program for specific FDT frame instructions.

9.4 Cable Connections

The PTQ-PDPMV1 module has the following communication connections on the module:

- One Ethernet port (RJ45 connector)
- One RS-232 Configuration/Debug port (DB9 connector)

9.4.1 Ethernet Connection

The PTQ-PDPMV1 module has an RJ45 port located on the front of the module labeled "Ethernet", for use with the TCP/IP network. The module is connected to the Ethernet network using an Ethernet cable between the module's Ethernet port and an Ethernet switch or hub.

Note: Depending on hardware configuration, you may see more than one RJ45 port on the module. The Ethernet port is labeled "Ethernet".

Warning: The PTQ-PDPMV1 module is NOT compatible with Power Over Ethernet (IEEE802.3af / IEEE802.3at) networks. Do NOT connect the module to Ethernet devices, hubs, switches or networks that supply AC or DC power over the Ethernet cable. Failure to observe this precaution may result in damage to hardware, or injury to personnel.

Important: The module requires a static (fixed) IP address that is not shared with any other device on the Ethernet network. Obtain a list of suitable IP addresses from your network administrator BEFORE configuring the Ethernet port on this module.

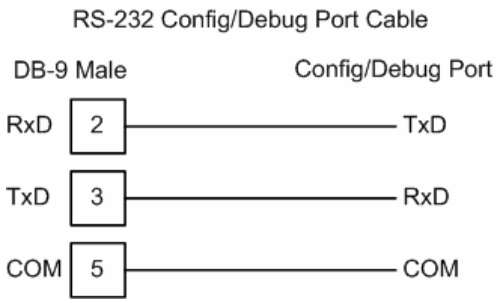
Ethernet Port Configuration: wattcp.cfg

The wattcp.cfg file must be set up properly in order to use a TCP/IP network connection. You can view the current network configuration using an ASCII terminal by selecting "@" (Network Menu) and "V" (View) options when connected to the Debug port.

```
# WATTCP.CFG FILE:
# Prosoft Technology.
  my_ip=192.168.0.100
# Default class 3 network mask
  netmask=255.255.255.0
# The gateway I wish to use
  gateway=192.168.0.1
```

9.4.2 RS-232 Configuration/Debug Port

This port is physically an RJ45 connection. An RJ45 to DB-9 adapter cable is included with the module. This port permits a PC based terminal emulation program to view configuration and status data in the module and to control the module. The cable for communications on this port is shown in the following diagram:



9.5 PROFIBUS Master Port

The following diagram has been imported from the PROFIBUS Master documentation. Note that the signals to reference are the D-Sub signals in the table

D-Sub (male)	Board to Board	Screw Terminal	Signal
Housing	1	5	Cable shield
1	4	-	-
2	7	-	-
3	6	4	B-Line
4	3	6	RTS
5	2	2	GND_BUS
6	8	1	+5V BUS (output)
7	9	-	-
8	5	3	A-Line
9	10	-	-

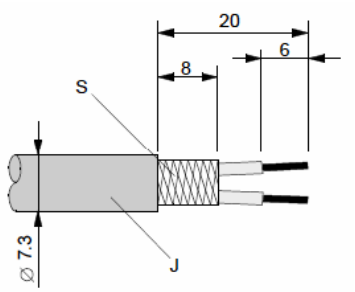
9.6 Constructing a Bus Cable for PROFIBUS DP

The bus cable for connecting PROFIBUS DP devices must be constructed by the user. A special PROFIBUS cable (twisted pair) is required here. This standard cable is available from various manufacturers and is a Belden part number of 3079A

To construct the cable, proceed as follows:

1 Cut the cable to the required length.

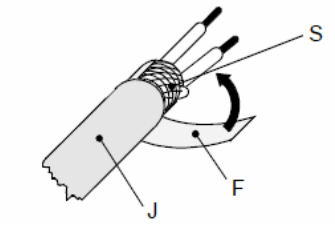
2 Prepare the cable ends as shown in the illustration (dimensions in mm):



J PVC Jacket
S Braided shielding

3 Remove the PVC jacket J to the indicated length.

4 Wrap the provided copper shielding F around the shield braiding S:

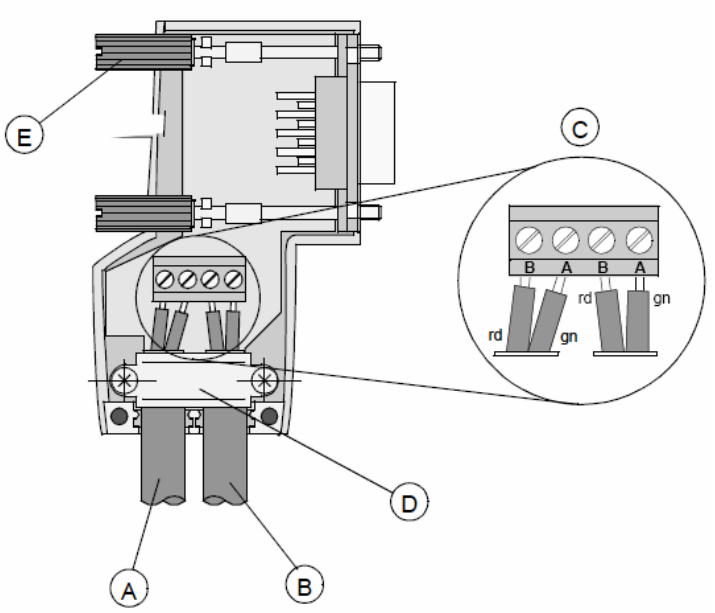


J PVC jacket
S Braided shielding
F Copper foil shielding
Additional foil can be obtained from 3M.

5 Plug the leads of the corresponding cable(s) into the terminals as shown:
o Green leads in terminal A
o Read lead in terminal B

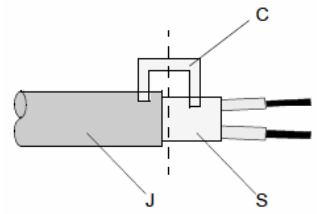
- **Note:** Do **not** tighten the corresponding screws yet.

Connection terminal assignment on the PROFIBUS DP:



A Incoming cable
B Outgoing cable
C Connection terminals (only once (B,A))
D Cable cleat for reliving tension
E Bus connector screws

6 Attach the cables with the provided cable cleat to create a robust shielded connection and to relieve any tension as shown:



J PVC Jacket
S Braided shielding with foil shielding
C Cable cleat

- **Note**: Half of the cable jacket must lie under the cable cleat!
  Pay attention to the cable cleat installation instructions.

7 Fasten the individual wires of the PROFIBUS cable to the terminals
8 Close the connector housing.

- **Note:** The shielding of both cables is connected internally with the metal housing of the connector.

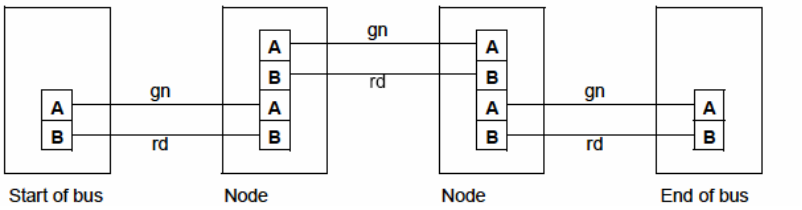
9 Complete the Central Shielding Measures (below) and grounding operations for the shielding before you connect the cable connector to the module.
10 Plug the PROFIBUS DP connector into the module and secure it with the screws.

Bus Begin and Bus End

The PROFIBUS connector with termination is required at the beginning and the end of the bus. These connectors emulate the line impedance.

It is recommended that at least one connector with diagnostics interface is used.

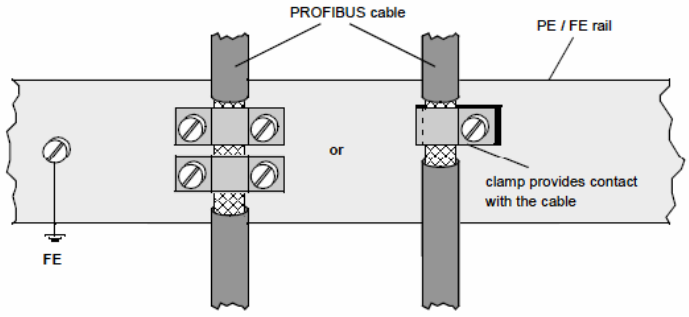
Wiring diagram for a PROFIBUS DP cable



Grounding and Shielding for Systems with Equipotential Bonding

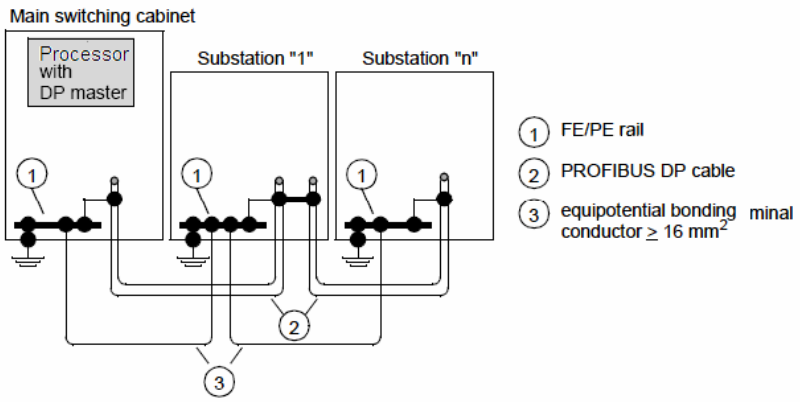
Each cable shield should be galvanically grounded with the earth using FE/PE grounding clamps immediately after the cable has been connected to the cabinet.

This example indicates the shielding connection from the PROFIBUS cable to the FE/PE rail.



Note: An equalization current can flow across a shield connected at both ends because of fluctuations in ground potential. To prevent this, it is imperative that there is potential equalization between all the attached installation components and devices.

This example indicates the system components and devices in a system with equipotential bonding.

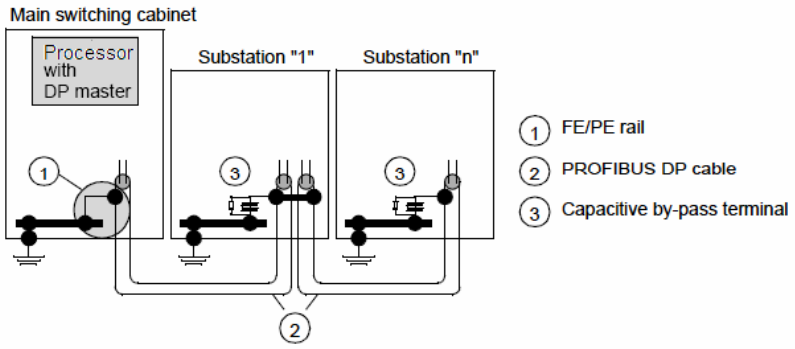


Grounding and Shielding for Systems without Equipotential Bonding

Note: Grounding and shielding is to be carried out the same as for systems **with** equipotential bonding.

If this is not possible because of system or construction specific reasons however, use distributed ground with a capacitive coupling of high frequency interference signals.

This representation shows distributed grounding with capacitive coupling.



9.7 Supported PROFIBUS Services

The following table lists all available services according to the PROFIBUS specification.

Service	PROFIBUS Version	Master Class 1		Master Class 2	
		Request	Response	Request	Response
DDLM_Data-Exchange	DPV0	Yes		No	
DDLM_Set_Prm	DPV0	Yes		No	
DDLM_Chk_cfg	DPV0	Yes		No	
DDLM Slave Diag	DPV0	Yes		No	
DDLM_Global_Control	DPV0	Yes		No	
DDLM_Get_Cfg	DPV0			Yes	
DDLM_Set_Slave_Add	DPV0			Yes	
DDLM_Read_Input	DPV0			No	
DDLM_Read_Output	DPV0			No	
DDLM_Get_Master_Diag	DPV0		Yes		
DDLM_Start_Seq	DPV0		No	No	
DDLM_Download	DPV0		No	No	
DDLM_Upload	DPV0		No	No	
DDLM_End_Seq	DPV0		No	No	
DDLM_Act_Param_Brct	DPV0		No	No	
DDLM_Act_Param	DPV0		No	No	
MSAC1_Read	DPV1	Yes			
MSAC1_Write	DPV1	Yes			
MSAL1_Alarm	DPV1		Yes		
MSAL1_Alarm_Ack	DPV1		Yes		
MSAC2_Initiate	DPV1			No	
MSAC2_Read	DPV1			No	
MSAC2_Write	DPV1			No	
MSAC2_DataTransport	DPV1			No	
MSAC2_Abort	DPV1			No	
Data_eXchange_Broadcast	DPV2	No			
Isochrone_mode (Takt sync)	DPV2	No			
Extended_Set_Prm (Subscriber)	DPV2	No			

9.8 Quantum to PTQ Communication Protocol

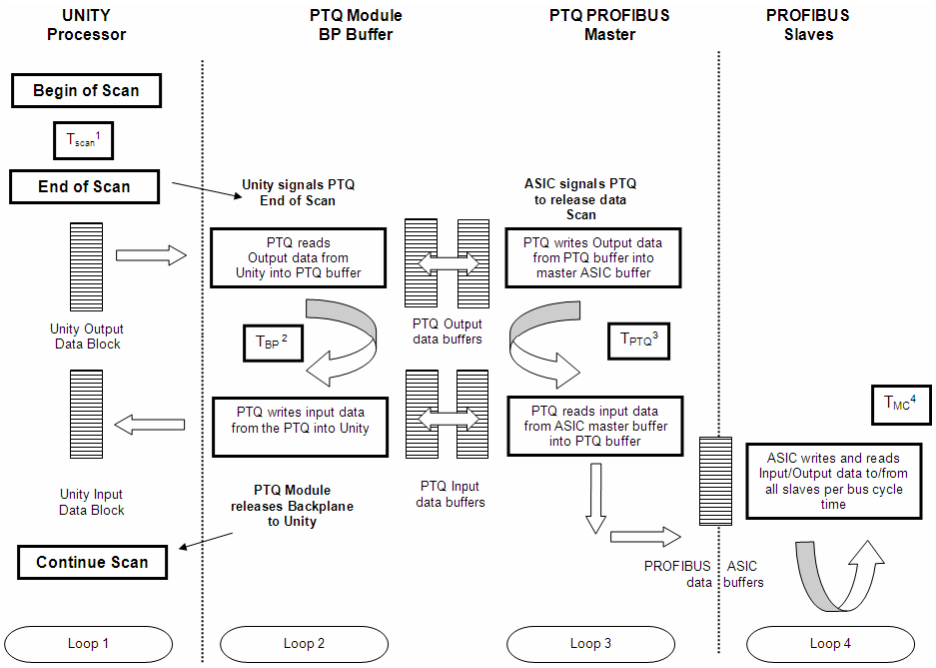
The vehicle utilized for transferring data between the PTQ module and the processor are two blocks of data:

- PTQ Input Data Block
- PTQ Output Data Block

Each of these data blocks (controlled by the PTQ) consists of a structure of data that provides for the movement of:

- Input Data image from PROFIBUS Slave devices
- Output Data image for writing to PROFIBUS Slave devices
- PTQ Module Configuration and Status (from PTQ to processor)
- PROFIBUS Messaging Mailbox commands (from processor to PTQ)
- PROFIBUS Messaging Mailbox responses (from PTQ to processor)

It is important to understand the process and flow of this data. The following illustration describes the mechanism in a block diagram overview. Several asynchronous data transfer loops occur simultaneously.


UNITY Processor
PTQ Module BP Buffer
PTQ PROFIBUS Master
PROFIBUS Slaves
Begin of Scan
Tscan 1
End of Scan
Unity signals PTQ End of Scan
ASIC signals PTQ to release data Scan
PTQ reads Output data from Unity into PTQ buffer
PTQ writes Output data from PTQ buffer into master ASIC buffer
Unity Output Data Block
PTQ Output data buffers
TBP 2
TPTQ 3
TMC 4
PTQ writes input data from the PTQ into Unity
PTQ reads input data from ASIC master buffer into PTQ buffer
ASIC writes and reads Input/Output data to/from all slaves per bus cycle time
Unity Input Data Block
PTQ Module releases Backplane to Unity
PTQ Input data buffers
Continue Scan
PROFIBUS data
ASIC buffers
Loop 1
Loop 2
Loop 3
Loop 4

PLC Scan Loop 1

The PLC processor performs cyclic program, backplane and network communication tasks for each PLC scan time. This time is referred to in this document as T_{SCAN} time. Refer to the Quantum processor manual for additional PLC cyclic task processing information.

Backplane Loop 2

When the processor reaches end of scan (EOS), the processor provides an interrupt to the PTQ. The PTQ locks out the processor and takes control of the

backplane for a period of time to transfer all input and output data described above. After the data is transferred the PTQ releases the backplane hook to the processor, and the processor continues with the next scan. The process repeats for every scan time. The time, T_{BP} backplane transfer time, is nominally 7.0ms for a complete transfer of data. This is maximum time for all 1984 input bytes and 1838 output bytes (this includes cyclic, acyclic, mailbox command data and input output status data). The user is able to control the number of input and output bytes within ProSoft Configuration Builder. The module supports 1536 bytes input data and 1536 bytes output cyclic data.

PTQ-PDPMV1 Main Loop 3

The PTQ transfers the data from the backplane buffer to the PROFIBUS master buffer in preparation to condition the data for slave type data consistency. This time, T_{PTQ}, PTQ process time, is nominally 45ms. During this loop, the PTQ module alternative locks the input and output data areas and exchanges the data. It also transfers acyclic mailbox data.

PROFIBUS Loop 4

The PROFIBUS cycle time is based on many factors, including:

- synchronization time,
- slave response time,
- idle time,
- bus baud rate

The cycle time "TMC" is calculated and added to processor scan time "T_{PTQ}" and "T_{BP}" time to arrive at the maximum response time of the PROFIBUS system.

9.9 Calculating System Response Time

Processor, PTQ module and PROFIBUS system response times are essentially dependent on the following factors:

- T_{SCAN}^{1} processor scan time (Loop1)
- T_{BP}^{2} max. BP transfer time (Loop 2)
- T_{PTQ}^{3} max. PTQ process time (Loop 3)
- T_{MC}^{4} time of PROFIBUS message cycle time (Loop 4)

PTQ max. T_{SR} (System Response Time) = $T_{SCAN}^{1} + T_{BP}^{2} + T_{PTQ}^{3} + T_{MC}^{4}$

TSR = 7.0ms + 45ms + 2.84ms + TSCAN1

where:

T_{BP}^{2} = ~7.0ms
T_{PTQ}^{3} = max. ~45ms
T_{MC}^{4} = 2.84ms (12PDPMV1s)

9.9.1 How to Calculate PROFIBUS Time – TMC4

First, a review of a few terms before getting into the details of calculating bus cycle times:

Bit-Time: To help simplify timing calculations, it is convenient to normalize the time units with respect to the baud rate by using units of Bit-Time (Tbit). One bit-time is the time it takes to transmit one bit and is the reciprocal of the transmission rate (baud rate). For example:

- 1 Tbit (Bit Time) at 12MB = 1/12000000bps = 83ns/bit
- 1 Tbit (Bit Time) at 1.5MB = 1/1500000bps = 667ns/bit

Sync-Time (T_{SYN})

The synchronization time is the minimum time a station must remain in the idle state before it can accept another request. For PROFIBUS DP, an idle state of 33Tbits (bit-time) must be present before every request telegram and this is called the sync-time.

Slave Reaction Time (T_{SDR})

The reaction time is the time it takes a slave to respond to a message. This time is often expressed as a minimum value (min T_{SDR}), or maximum value (max T_{SDR}). Min T_{SDR} is set within the parameterization telegram during startup. Max T_{SDR} varies with the transmission rate and is specified at the supported baud rates within the device GSD file. For PROFIBUS DP, this value may range from a minimum of 11Tbits (min T_{SDR} default) to a maximum of 255Tbits.

Initiator Delay Time (T_{SDI})

T_{SDI} refers to the station delay of the initiator of a request or token frame (the master).

Initiator Idle Time (T_{ID}^1)

After receiving the last character of a telegram, the initiator must wait this amount of time until it sends the next telegram. The idle time (T_{ID}^1) is the time between transmission of the last bit of a frame (no acknowledge) and the transmission of the first bit of the next frame. It is at least the sync time (T_{SYN}), plus some safety margin (T_{SM}), but is also calculated as the maximum of these three values: T_{SYN} + T_{SM}, min T_{SDR}, or T_{SDI} (station delay of telegram initiator). The addition of safety margin (T_{SM}) is very important at high baud rates.

Minimum Slave Interval

The minimum slave interval is the minimum time that must expire between two slave polling cycles in which a slave can exchange data with the master. To permit the slave station to be able to respond during every data cycle, it controls the bus cycle with this parameter. It is defined in the slave's GSD file via the parameter Min_Slave_Interval, which is specified as a 16 bit factor of 100us (Min_Slave_Interval = 1 is 100us). On some older equipment, the PROFIBUS link was implemented in software (as opposed to within the slave ASIC) and a typical value was about 2ms. On newer equipment with modern ASICs, values down to 100μs can be achieved.

9.9.2 Calculating System Reaction Time

A simplified calculation of system reaction time for a PROFIBUS DP system is derived from the following parameters:

- T_{SDR} (Station Reaction Time).
- The Transmission (Baud) Rate.
- The Net Data Length specified.
- Min_Slave_Interval (min time between two slave polling cycles)

Example: One-master and x slaves are connected via PROFIBUS DP. We will use the maximum available bytes of 1536 bytes of output data and 1536 bytes of input data are to transfer to maximum number slaves allowed using 1536 input and output bytes at 12PDPMV1s. Each slave utilizes an SPC3 ASIC. To calculate the relative bus cycle time for this network: Let T_{MC} = Time of 1 telegram cycle (request telegram + T_{SDR} + slave response). Let T_{BC} = Time of 1 bus cycle (the sum of all telegram cycles).

Given:

T_{SYN} = 33 TBits (Bus idle time or PROFIBUS Sync-Time)
T_{ID}^1 = 75 TBits (SPC3 bus idle time, at 1.5MB T_{ID}^1 = 36 TBit).
T_{SDR} = 30 TBits typical for baud rates ≥ 1.5MB (SPC3 ASIC).
Min_Slave_Interval = 1 (100μs, from slave GSD file).

Calculate:

1 Tbit (Bit Time) at 12MB = 1/12000000bps = 83ns/bit

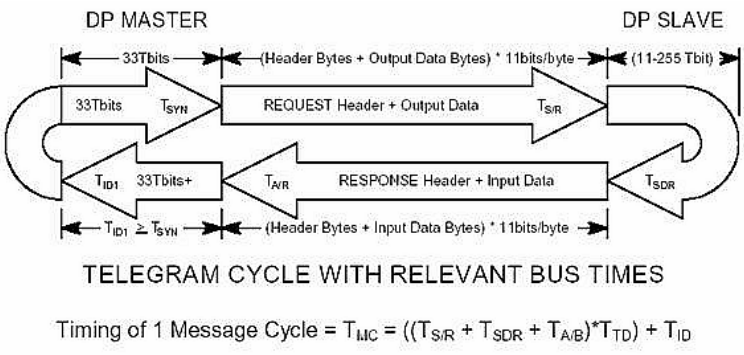
In data exchange mode, a telegram header consists of only 9 character bytes. If we include the bits of the character frame, there are 11 bits for every character byte (Start Bit + 8bits/char + Stop Bit + Parity). Because only 1 master is present, we can ignore the token hold time of token telegrams. Thus, the basic time required by one telegram cycle (not including data) is obtained by adding the relevant bus times and the time to transmit the telegram header as follows:

T_{MC}^4 (in TBits) = 2 * 9(header byte length) * 11bits/byte + T_{SDR} + T_{SYN} + Tid^1
T_{MC}^4 = 198 bits + 30 bits + 33 bits + 75 bits = 336 Tbits
T_{MC4} (μs) = 336Tbits * 83ns/Tbit = 28μs

Thus, 28μs is the basic time required by the telegram header including the bus times without accounting for the data. For our example, we must include the data (1536 bytes Output + 1536 bytes Input – maximum bytes for a PTQ-PDPMV1). The time for a single telegram cycle with this data included is:

TMC4 = [336Tbits] + amount of net data = 336 + [(1536bytes Output + 1536bytes Input)*(11bits/byte)] = 34128 Tbits
TMC4 = 34128 * 83ns/bit
TMC4 = 2.84ms

To simplify this calculation, you can assume that a basic transfer time of 28us plus approximately 1μs per DU data byte (actually 0.83μs/byte) is required to complete a telegram cycle. The following illustration gives an overview of the dominant bus times in a telegram cycle (assuming no interference or repetitions).



Note that the slave has a Min_Slave_Interval of 100µs and this dominates the bus timing for one telegram cycle. However, the Min_Slave_Interval is 100µs between two polling cycles at the same station. If you have at least 3 stations present, then the actual transmission time at 12MB will become the determining time factor for the bus cycle rather than the Min_Slave_Interval. Refer to the EN50170 standard for a more detailed calculation of transmission time.

Reference: INTRODUCTION TO PROFIBUS DP, ACROMAG INCORPORATED.

9.10 Using Multiple PTQ-PDPMV1 Modules with Concept

If your application requires more than one PTQ-PDPMV1 module for a single Quantum processor, you must manually merge the exported DTY files for each module into a single DTY file.

The only difference between the DTY files for each module is the PROFIBUS data types (input and output) that define the data associated to configured slaves. If you have already exported the processor files for the first modules (C:\project\DFB), repeat the following steps for each additional module to include in the Concept project.

1. Export the new processor files to a different folder (for example C:\temp).
2. Use a text editor such as Notepad.exe to open the exported DTY file. Select and copy the PROFIBUS "DATAF" data type definitions. These are the last two data types defined in the DTY file. The name of these data types will vary depending on the module name you entered, but they will always have "_IN_DATAF" and "OUT_DATAF" suffixes. For example, if the module name was configured as "PTQPDPMV1", select and copy the following data types:

```
TYPE PTQPDPMV1_IN_DATAF:
...
...
END_TYPE
TYPE PTQPDPMV1_OUT_DATAF:
...
...
END_TYPE
```

3. Open the DTY file in C:\project\DFB and paste the data types at the end of the file. Save and close the file.

With this procedure, you will obtain a final merged DTY file for all PTQ-PDPMV1 modules for your application.

9.11 Frequently Asked Questions

9.11.1 How do I configure the module?

The PTQ-PDPMV1 requires a simple text based configuration file to make it operational.

9.11.2 Is a .MDC available for configuration of the Module?

Yes. The CD-ROM that ships with the module should have a version for both Concept 2.5 and 2.6 in the PTQ-PDPMV1 directory.

9.11.3 Does the module work in a remote rack?

The module is designed to be located in the chassis with the PLC and will not operate in a remote chassis. If your application requires remote placement of the communication device you should investigate the other members of the ProSoft family such as the ProLinx gateway solutions.

9.11.4 Can I use the module in a hot backup system?

The PTQ-PDPMV1 module supports the 140CPU67160 Hot Standby processor. Refer to ***Hot Standby Support*** (page 137) for setup and configuration instructions.

Support, Service & Warranty

ProSoft Technology, Inc. survives on its ability to provide meaningful support to its customers. Should any questions or problems arise, please feel free to contact us at:

Internet	Web Site: http://www.prosoft-technology.com/support E-mail address: support@prosoft-technology.com
Phone	+1 (661) 716-5100 +1 (661) 716-5101 (Fax)
Postal Mail	ProSoft Technology, Inc. 1675 Chester Avenue, Fourth Floor Bakersfield, CA 93301

Before calling for support, please prepare yourself for the call. In order to provide the best and quickest support possible, we will most likely ask for the following information:

1. Product Version Number
2. System architecture
3. Module configuration and contents of configuration file, if the module requires one.
4. Module Operation
   - Configuration/Debug status information
   - LED patterns
5. Information about the processor and user data files as viewed through the processor configuration software and LED patterns on the processor
6. Details about the serial devices interfaced

An after-hours answering system allows pager access to one of our qualified technical and/or application support engineers at any time to answer the questions that are important to you.

Module Service and Repair

The PTQ-PDPMV1 device is an electronic product, designed and manufactured to function under somewhat adverse conditions. As with any product, through age, misapplication, or any one of many possible problems the device may require repair.

When purchased from ProSoft Technology, Inc., the device has a 1 year parts and labor warranty (3 years for RadioLinx) according to the limits specified in the warranty. Replacement and/or returns should be directed to the distributor from whom the product was purchased. If you must return the device for repair, obtain an RMA (Returned Material Authorization) number from ProSoft Technology, Inc. Please call the factory for this number, and print the number prominently on the outside of the shipping carton used to return the device.

General Warranty Policy – Terms and Conditions

ProSoft Technology, Inc. (hereinafter referred to as ProSoft) warrants that the Product shall conform to and perform in accordance with published technical specifications and the accompanying written materials, and shall be free of defects in materials and workmanship, for the period of time herein indicated, such warranty period commencing upon receipt of the Product. Limited warranty service may be obtained by delivering the Product to ProSoft in accordance with our product return procedures and providing proof of purchase and receipt date. Customer agrees to insure the Product or assume the risk of loss or damage in transit, to prepay shipping charges to ProSoft, and to use the original shipping container or equivalent. Contact ProSoft Customer Service for more information.

This warranty is limited to the repair and/or replacement, at ProSoft's election, of defective or non-conforming Product, and ProSoft shall not be responsible for the failure of the Product to perform specified functions, or any other non-conformance caused by or attributable to: (a) any misuse, misapplication, accidental damage, abnormal or unusually heavy use, neglect, abuse, alteration (b) failure of Customer to adhere to ProSoft's specifications or instructions, (c) any associated or complementary equipment, software, or user-created programming including, but not limited to, programs developed with any IEC1131-3 programming languages, 'C' for example, and not furnished by ProSoft, (d) improper installation, unauthorized repair or modification (e) improper testing, or causes external to the product such as, but not limited to, excessive heat or humidity, power failure, power surges or natural disaster, compatibility with other hardware and software products introduced after the time of purchase, or products or accessories not manufactured by ProSoft; all of which components, software and products are provided as-is. In no event will ProSoft be held liable for any direct or indirect, incidental consequential damage, loss of data, or other malady arising from the purchase or use of ProSoft products.

ProSoft's software or electronic products are designed and manufactured to function under adverse environmental conditions as described in the hardware specifications for this product. As with any product, however, through age, misapplication, or any one of many possible problems, the device may require repair.

ProSoft warrants its products to be free from defects in material and workmanship and shall conform to and perform in accordance with published technical specifications and the accompanying written materials for up to one year (12 months) from the date of original purchase (3 years for RadioLinx products) from ProSoft. If you need to return the device for repair, obtain an RMA (Returned Material Authorization) number from ProSoft Technology, Inc. in accordance with the RMA instructions below. Please call the factory for this number, and print the number prominently on the outside of the shipping carton used to return the device.

If the product is received within the warranty period ProSoft will repair or replace the defective product at our option and cost.

Warranty Procedure: Upon return of the hardware product ProSoft will, at its option, repair or replace the product at no additional charge, freight prepaid, except as set forth below. Repair parts and replacement product will be furnished on an exchange basis and will be either reconditioned or new. All replaced product and parts become the property of ProSoft. If ProSoft determines that the Product is not under warranty, it will, at the Customer's option, repair the Product using then current ProSoft standard rates for parts and labor, and return the product freight collect.

Limitation of Liability

EXCEPT AS EXPRESSLY PROVIDED HEREIN, PROSOFT MAKES NO WARRANT OF ANY KIND, EXPRESSED OR IMPLIED, WITH RESPECT TO ANY EQUIPMENT, PARTS OR SERVICES PROVIDED PURSUANT TO THIS AGREEMENT, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER PROSOFT OR ITS DEALER SHALL BE LIABLE FOR ANY OTHER DAMAGES, INCLUDING BUT NOT LIMITED TO DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION IN CONTRACT OR TORT (INCLUDING NEGLIGENCE AND STRICT LIABILITY), SUCH AS, BUT NOT LIMITED TO, LOSS OF ANTICIPATED PROFITS OR BENEFITS RESULTING FROM, OR ARISING OUT OF, OR IN CONNECTION WITH THE USE OR FURNISHING OF EQUIPMENT, PARTS OR SERVICES HEREUNDER OR THE PERFORMANCE, USE OR INABILITY TO USE THE SAME, EVEN IF ProSoft OR ITS DEALER'S TOTAL LIABILITY EXCEED THE PRICE PAID FOR THE PRODUCT.

Where directed by State Law, some of the above exclusions or limitations may not be applicable in some states. This warranty provides specific legal rights; other rights that vary from state to state may also exist. This warranty shall not be applicable to the extent that any provisions of this warranty are prohibited by any Federal, State or Municipal Law that cannot be preempted. Contact ProSoft Customer Service at +1 (661) 716-5100 for more information.

RMA Procedures

In the event that repairs are required for any reason, contact ProSoft Technical Support at +1 661.716.5100. A Technical Support Engineer will ask you to perform several tests in an attempt to diagnose the problem. Simply calling and asking for a RMA without following our diagnostic instructions or suggestions will lead to the return request being denied. If, after these tests are completed, the module is found to be defective, we will provide the necessary RMA number with instructions on returning the module for repair.

Index

%

%SW60 Hot Standby Command Register • 156
%SW61 Hot Standby Status Register • 158

.

-.ASC files • 77
-.DTY file • 78
-.TXT file • 78

A

About the PROFIBUS Protocol • 193
Accessing PROFIBUS Data • 92
Accessing Status Data • 93
Active and Passive HSBY Master Status data High byte – bits 8 to 15 • 155
Active and Passive HSBY Master Status data Low byte – bits 0 to 7 • 154
Acyclic Read Mailbox • 64, 100
Acyclic Write Mailbox • 65, 101
Alarm Mailbox • 65, 102

B

Back Up the Project • 36, 39, 41
Basic Troubleshooting Steps • 173
Before You Begin • 74
Behavior of segment coupler 2 • 170
Build the Project • 50
Bus Access • 195
Bus Parameters • 162, 165

C

Cable Connections • 217
Calculate Checksums • 35
Calculating System Reaction Time • 227
Calculating System Response Time • 225
Can I use the module in a hot backup system? • 229
Coldboot Mailbox • 71, 107
Commissioning of communication with the SK1 segment coupler • 167
Commissioning of communication with the SK2 segment coupler • 169
Completing the Example Configuration • 24
Conditions for Switchover • 151
Configuration • 60, 95
Configuration Validation & SETCRC Function Block • 35, 78, 94
Configure the Memory Size for the Processor • 48
Configure the Module with ProSoft Configuration Builder • 15
Configure the Processor • 38
Configure the PROFIBUS Master • 27
Configure the PROFIBUS Slaves • 29
Configuring the Module • 15, 48
Configuring the Processor with Concept • 34, 73
Configuring the Processor with ProWORX 32 • 109
Configuring the Processor with Unity Pro XL • 34, 39
Connect to the ProTalk Configuration/Debug Port • 11
Constructing a Bus Cable for PROFIBUS DP • 218
Crossed status information • 158
Crossed Status Information • 150

D

Database Mismatch • 160
Deployment Checklist • 9
Derived Function Blocks Overview • 57
Details for calculating the TWD parameter • 171
Determining the parameter TWD • 170
Diagnostics and Troubleshooting • 7, 173
Does the module work in a remote rack? • 229
Download the Project to the Module • 35
Download the Project to the Processor • 51
Downloading the Ethernet Configuration to the Module • 26
DPV1 Error Codes • 136

E

Effect on System Scan Time • 139
Error Codes • 135
Ethernet Connection • 217
Ethernet Port Configuration
wattcp.cfg • 217
Exiting the Program • 185
Export the Processor Memory Map • 33

F

FDL Layer Access • 161
Features and Benefits • 191
File Locations • 37
Frequently Asked Questions • 229
Function Blocks Operation Overview • 54
Functional Overview • 7, 193
Functional Specifications • 192

G

General Overview • 193
General Specifications • 191
GetConfiguration Mailbox • 67, 103
GetDiagnostics Mailbox • 67, 103
GetLiveList Mailbox • 68, 104
Glossary of Terms • 143
Guide to the -PDPMV1 User Manual • 7

H

Hardware Requirements (Recommended) • 210
Hardware Specifications • 192
Highest Station Address (HSA) • 163
Hot Standby Overview • 137
Hot Standby States • 144
Hot Standby Support • 137, 191, 229
How do I configure the module? • 229
How the PTQ detects when a switch over is detected • 158
How to Calculate PROFIBUS Time – TMC4 • 226
HSBY Master GSD-file • 163
HSBY Nr of Active slaves byte • 155
HSBY Nr of Passive slaves byte • 155
HSBY Operating Modes • 143
HSBY State vs. Master Operation Mode • 146

I

Identical Configurations • 137
Import the Functional Module • 39
Information for Concept Version 2.6 Users • 75
Information for determining the Watch Dog Time TWD • 169
Input Block Status Data Details • 22, 93, 201
Input Byte Swap • 22
Input Data Size • 21
Input Data View Menu • 185
Input Start Register • 21
Input/Output PTQ-PDPMV1 Image table • 196
Input/Output PTQ-PDPMV1 Image with Mailbox messaging and slave Diagnostics table • 197
Input/Output PTQ-PDPMV1 Image without Mailbox messaging and with slave Diagnostics table • 198
Input/Output PTQ-PDPMV1 Image without Mailbox messaging and Without Slave Diagnostics table • 199
Install ProSoft Configuration Builder Software • 10
Install the GSD Files • 28
Install the Module • 10
Install the ProTalk Module in the Quantum Rack • 10
Installation • 210
Installing MDC Configuration Files • 75
Introduction • 207
Is a .MDC available for configuration of the Module? • 229

K

Keystrokes • 180

L

LED indicators • 163
LED Status Indicators - Front of PTQ module • 7, 174
Limitations • 143

M

Mailbox Message - Acyclic Data Read - Class 1 • 126
Mailbox Message - Acyclic Data Write - Class 1 • 127
Mailbox Message - Alarm Indication • 61, 129
Mailbox Message - Get Live List • 124
Mailbox Message - Get Slave Configuration • 121
Mailbox Message - Get Slave Diagnostics • 119
Mailbox Message - Set Operating Mode • 131
Mailbox Message - Set Slave Address • 122
Mailbox Message - Set Slave Mode • 116
Mailbox Message Queuing • 113
Mailbox Messaging • 22, 53, 56, 59, 64, 100, 113
Mailbox Messaging Error Codes • 68, 120, 122, 124, 134
Mailbox Overview • 64, 100
Main Menu • 180
Mapping the Backplane Extensions • 143
Master address (TS) • 163
Master/Slave Communication Phases • 195
Master/Slave Polling • 195
Message Information • 117, 120, 122, 123, 125, 126, 128, 134
Modicon Quantum Hot Standby with Unity and IEC Logic • 139
Module Status Indicators • 177

N

Navigation • 179
New HSBY Input Status Data Word Details • 154
Normal Operation • 196

O

Offline to Primary (2) • 145
Offline to Standby (1) • 145
Opening the Input Data View Menu • 181
Opening the Output Data View Menu • 182
Output Byte Swap • 22
Output Data Size • 21
Output Data View Menu • 186
Output Start Register • 21
Overview • 60, 73, 95, 139, 142

P

Parameter Calculation • 165
PCB Master Configuration Software • 164
Performance Considerations • 140
Ping Message • 146
Ping Message Structure • 149
Ping Sequence • 148
PLC HSBY Command Register • 156
PLC HSBY Status Register • 157
Please Read This Notice • 2
Preconditions • 147
Primary and Standby Controllers • 138
Primary to Offline (4) • 145
Product Specifications • 7, 189
PROFIBUS comDTM • 207
PROFIBUS DP Architecture • 194
PROFIBUS DP time behavior • 166
PROFIBUS Master Indicators • 178
PROFIBUS Master Port • 218
ProSoft Configuration Builder (PCB) HSBY Option Functionality • 164
ProSoft Technology product availability • 207
PTQ Input and Output Data Blocks • 196
PTQ Input Data Block • 22, 24, 152, 196
PTQ Link Message • 150
PTQ Output Data Block • 206
PTQ-PDPMV1 Configuration / Debug Port Note • 13
PTQ-PDPMV1 Hot Standby Considerations • 143
PTQ-PDPMV1 HSBY Diagnostic Data • 152
PTQ-PDPMV1 Master Bus Properties when using a P&F DP/PA Segment Coupler • 166
PTQ-PDPMV1 Operation • 152

Q

Quantum to PTQ Communication Protocol • 224
Queue Timeouts • 114
Quick Start • 211

R

Receiving Mailbox Message Responses from module • 132
Redisplaying the Menu • 181, 186
Reference • 7, 189
Return Codes • 120, 122, 124, 127, 129, 131, 134
Returning to the Main Menu • 186, 187
RS-232 Configuration/Debug Port • 218

S

Set Module Parameters • 19
Set Up the Project • 17
Set_Param (SAP61) • 32
SetOperatingMode Mailbox • 69, 105
SetSlaveAddress Mailbox • 69, 105
SetSlaveMode Mailbox • 70, 106
Setting Up the Modicon Quantum Hot Standby with Unity System • 142
Setup General Unity Pro XL Project Settings • 46
Slave Diagnostics • 22
Slave List Structure for Offset Words 13, 21 and 29 • 202
Slave status • 159
Slave watchdog time (WD_Factor1&2) • 162
Slot Number • 20
Software Requirements (Minimum) • 210
Special Function Mailbox Messaging Commands • 59, 114
Specific Input Pins • 62, 97
Specific Output Pins • 62, 97
Standby to Offline (5) • 146
Standby to Primary (3) • 145
Start Here • 7, 9
Start/Stop Slaves Mailbox • 106
StartStopSlaves Mailbox • 70
Start-up • 148
State Description • 144
Step 1 - Exporting the Files from PCB • 73, 76
Step 2 - Convert the Function Blocks • 74, 78
Step 3 - Setup the Concept Project • 74, 81
Step 4 - Import the Variables • 74, 83
Step 5 - Create the Function Block Instances • 74, 86
Step 6 - Download the Concept Project • 74, 92
Support, Service & Warranty • 7, 231
Supported PROFIBUS Services • 223
System Components • 138
System Requirements • 210

T

The Configuration/Debug Menu • 178
Time for Switchover • 162
Token Passing • 195

Token rotation time (TTR) • 162
Transition Description • 145
Trigger (BYTES) • 61
Trigger Register • 96

U

Understanding System Scan Time in Modicon Quantum Hot Standby with Unity Systems • 139
Unsupported functions • 163
Update the Ethernet Settings • 25
Updating Checksum Values - UnityPro XL • 35, 44
Using a Control Word to Reboot the Module • 72, 108
Using Mailbox Function Blocks • 60, 95
Using Multiple PTQ-PDPMV1 Modules with Concept • 78, 228
Using the Concept Project • 92
Using the Derived Function Blocks • 57
Using the Diagnostic Window in ProSoft Configuration Builder • 178

V

Verify Communication between the Processor and the Module • 51
Viewing Block Transfer Statistics • 181
Viewing Control Registers • 184
Viewing Data in ASCII (Text) Format • 186, 187
Viewing Data in Decimal Format • 185, 186
Viewing Data in Floating Point Format • 186, 187
Viewing Data in Hexadecimal Format • 185, 186
Viewing Ethernet NIC Configuration • 184
Viewing Fieldbus Data • 182
Viewing Module Configuration • 181
Viewing Module Status • 182
Viewing the Next Page of Data • 185, 186
Viewing the Previous Page of Data • 185, 186
Viewing the WATTCP.CFG File • 185
Viewing Version Information • 182

W

What is DTM? • 209
What is FDT? • 207
What is PROFIBUS? • 209
Why use PROFIBUS comDTM? • 207
Word offset 60 HSBY Active Master Status data • 154
Word offset 61 HSBY Passive Master Status data • 154

Y

Your Feedback Please • 2